UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Grado en Ingeniería Informática

Aplicación web para una tienda de donaciones

Autor: Omar Anibal Garcia Rios

Fecha de defensa: 5 de julio del 2019

Universidad: Universitat Politècnica de Catalunya (UPC) – BarcelonaTech

Centro: Facultat d'Informàtica de Barcelona (FIB)

Titulación: Grado en Ingeniería Informática

Especialidad: Ingeniería del Software

Ponente y Departamento del Ponente: Cristina Gomez Seoane, Ingeniería de Servicios y Sistemas de Información

Director e Institución del Director: Pontus Österberg, Prototyp Stockholm AB

Modalidad: B

Empresa: Prototyp Stockholm AB

Resumen

El proyecto "Diseño e implementación de una aplicación web para una tienda de donaciones" tiene como objetivo desarrollar una aplicación web que permita a los usuarios comprar donaciones en forma de regalo y que los destinatarios de éstos reciban una experiencia personalizada a través de un cuestionario educativo y un vídeo mensaje que tiene como finalidad sensibilizar y alentar sobre problemas actuales.

El desarrollo de este proyecto se desarrollará usando una metodología ágil flexible, que estará dividida en seis iteraciones de dos semanas. A lo largo de ésta memoria se irá explicando el proceso de desarrollo de forma cronológica, partiendo de conceptos aprendidos durante toda la carrera y mostrando una evolución de conocimientos.

Resum

El projecte "Disseny i implementació d'una aplicació web per a una botiga de donacions" té com a objectiu desenvolupar una aplicació web que permeti als usuaris comprar donacions en forma de regal i que els destinataris d'aquests rebin una experiència personalitzada a través d'un qüestionari educatiu i un vídeo missatge que té com a finalitat sensibilitzar i encoratjar sobre problemes actuals.

El desenvolupament d'aquest projecte es desenvoluparà utilitzant una metodologia àgil flexible, que estarà dividida en sis iteracions de dues setmanes. Al llarg d'aquesta memòria s'anirà explicant el procés de desenvolupament de forma cronològica, partint de conceptes apresos durant tota la carrera i mostrant una evolució de coneixements.

Abstract

The project "Design and implementation of a web application for a donation shop" aims to develop a web application that allows users to buy donations in the form of gifts and that the recipients of these receive a personalized experience through an educational questionnaire and a video message that aims to raise awareness and encourage current problems.

The development of this project will be developed using a flexible agile methodology, which will be divided into six iterations of two weeks. Throughout this report, the development process will be explained chronologically, starting from concepts learned throughout the career and showing an evolution of knowledge.

Agradecimientos

A mi ponente Cristina Gómez Seoane, que me ha guiado durante el desarrollo de éste TFG.

A mi director Pontus Österberg, que me ha dado la oportunidad de realizar el TFG en su empresa.

A mis compañeros de la empresa, que me han dado buenos consejos durante las horas del almuerzo.

A mi familia, por su apoyo incondicional.

Índice

1. Introducción y contextualización

Este es un proyecto de la especialidad de Ingeniería del Software con modalidad B, es decir, se realiza a través de un convenio entre la universidad y una empresa que he escogido. La empresa donde haré el TFG[1] se llama Prototyp SE[2] y es una agencia que se dedica principalmente a realizar prototipos de aplicaciones tanto web como móvil para clientes.

La empresa me propuso varios proyectos en los que estaban interesados y me decanté por la realización de una tienda online que ofrece la posibilidad de realizar donaciones relacionadas con el medio ambiente en nombre de otras personas y que éstos reciban un regalo personalizado (que será una carta que le permitirá acceder a un apartado especial en la aplicación web) que además enseñe sobre problemas actuales. Por lo tanto, el objetivo principal del proyecto es ofrecer una solución que extiendan las donaciones hacia proyectos medioambientales, donde el usuario que reciba el regalo de parte del donante sienta realmente que está obteniendo algo único.

1.1. Formulación del problema

Actualmente hay muchas celebraciones en las que es tradición realizar un regalo a otra persona como puede ser un cumpleaños o un regalo de boda. Hay veces en las que las personas no saben qué regalar en estas situaciones, por lo que muchas veces se realizan regalos que no son a gusto del receptor y acaban olvidados, cogiendo polvo en armarios que no se abren en mucho tiempo.

De hecho, podemos decir que estamos en una era de consumismo en la que muchas veces solo nos importa comprar cosas sin importar las consecuencias. Y es que muchas de estas compras son hechas sin sentido y pueden tener consecuencias graves en el medio ambiente si se mira desde una perspectiva en la que no sólo estamos nosotros, está toda una ciudad, todo un país.

Siguiendo un poco más el tema del medio ambiente, hoy en día el modelo de la economía de la sociedad actual se basa generalmente en la extracción, manufacturación, producción, distribución, compra y desecho de materias primas. De hecho voy a citar el extracto de un artículo interesante que leí en una web sobre sostenibilidad que refleja los problemas de éste modelo de economía [1]:

"*Durante la extracción de recursos naturales se obtienen materias primas y energía de la naturaleza para producir bienes y servicios. Muchos de estos recursos no son renovables o se regeneran muy lentamente, lo que supone un problema doble: por un lado estamos alternando los ciclos o la capacidad de regeneración de algunos recursos , como por ejemplo el ciclo del agua. Y por otro lado estamos produciendo materias primas y energía de manera muy contaminante; por ejemplo, con la quema de combustibles fósiles.*"

[1] Sigla correspondiente a: Trabajo de Fin de Grado.
[2] https://prototyp.se

De tal modo que en este TFG se va a realizar un proyecto que intenta lidiar con estos dos problemas. Por un lado una solución para regalos innecesarias que no aportan nada y por otro, aportar un pequeño grano de arena para mejorar el medio ambiente.

1.2. Estado del arte

Para situar el proyecto que voy a realizar dentro de un marco de soluciones ya existentes en el mercado, he realizado un estudio de mercado teniendo en cuenta la finalidad de la aplicación, para ver si hay otras iguales que ofrezcan lo mismo que la aplicación en la que tengo planeado trabajar.

1.2.1. Estudio del mercado

Para tener una idea general del mercado actual, he buscado aplicaciones similares a lo que tengo en mente realizar a través del buscador de Google[3].

Algomasqueunregalo[4]

Es una web donde los usuarios pueden donar a proyectos de varios tipos. Proyectos que van desde suministrar material educativo hasta contribuir a familias de Guatemala. Al realizar estas donaciones puedes escoger un regalo para otra persona. Estos regalos pueden ser: una tarjeta en papel, pegatina para enganchar a un regalo o una *eCard* (tarjeta electrónica) en el correo del destinatario. Cada uno de estos regalos contiene una imagen de acuerdo con la donación escogida y una texto escrito por el donante que será enviado a su dirección para que lo pueda entregar en mano al receptor.

Unicef[5]

La web de Unicef tiene un apartado en su sitio web donde se pueden hacer 'regalos azules'. Estos regalos se tratan de donaciones que se compran en forma de paquetes. Cada paquete indica a dónde irá destinado el dinero. Por ejemplo tenemos: 'kit de supervivencia', donde el dinero comprado se destina a comprar sobres de alimentos terapéuticos para tratar la desnutrición aguda. Una vez se compra el paquete escogido, puedes personalizar una tarjeta que será enviada al domicilio que quieras.

Heifer[6]

Esta tienda online tiene la interfaz sobrecargada de múltiples opciones para donar. Hay donaciones de todo tipo y te permite elegir que cantidad de dinero quieres donar. Permite pagos únicos y suscripciones mensuales. Permite personalizar una carta con el texto que tú elijas y ofrece dos opciones, descargar un documento con la carta para imprimir y dárselo a quien quieras o también está la opción de enviar una tarjeta electrónica al receptor del regalo.

[3] https://google.es
[4] https://algomasqueunregalo.oxfamintermon.org
[5] https://tienda.unicef.es
[6] https://www.heifer.org

Globalgiving[7]

Es una web que cuenta con una cantidad enorme de proyectos de todo tipo organizados en una lista. El funcionamiento de ésta es un poco diferente, ya que tiene una apartado donde se pueden comprar una tarjeta con saldo que se puede gastar en la web. De manera que los donantes compran una tarjeta que pueden personalizar y se lo entregan a los receptores en mano. Los receptores entonces pueden usar la tarjeta para hacer donaciones en la web.

1.2.2. Conclusiones del estudio de mercado

A modo de extraer conclusiones, he hecho una tabla con distintas funcionalidades que debería tener nuestra aplicación, pero antes voy a nombrarlas:

- Donación variable: Permite al donante apoyar a un proyecto libremente sin un precio fijo.
- Proyectos medioambiente: La aplicación permite apoyar a proyectos relacionados con el medio ambiente.
- Enseñanza interactiva: El receptor del regalo aprende de manera interactiva sobre el tema del cual se ha hecho una donación.
- Personalizable con vídeo: El donante puede personalizar el regalo del receptor con un vídeo mensaje subido por él.

	Algomasqueunregalo	Unicef	Heifer	Globalgiving	GiftOfCharity (nuestra)
Donación variable	✘	✘	✔	✔	✔
Proyectos medioambiente	✔	✘	✔	✔	✔
Enseñanza interactiva	✘	✘	✘	✘	✔
Personalizable con vídeo	✘	✘	✘	✘	✔

Tabla 1. Comparación de aplicaciones según funcionalidades que debería tener el proyecto.

Como podemos ver, la aplicación que voy a construir va a diferenciarse en que el receptor de los regalos van a obtener una manera de aprender sobre la donación de manera interactiva, de manera que realmente van a acabar aprendiendo algo.

Además, ninguna de las webs antes dichas tiene la opción de poder subir un vídeo mensaje, nuestro proyecto tiene como finalidad presentar una experiencia puramente digital, mientras que éstas otras se basan más en lo físico (postales con imagenes).

[7] https://www.globalgiving.org

1.3. Actores implicados

En este apartado voy a situar a lo principales actores implicados de la aplicación (también llamados *stakeholders*). Los actores implicados son las personas que se verán afectadas por la aplicación, es decir, todas las personas que de un modo u otro tendrán relación con el proyecto.

1.3.1. Ponente TFG

Cristina Gomez Seoane será mi ponente del trabajo de fin de grado que me guiará en todo el proceso de la documentación del proyecto.

1.3.2. Empresa Prototyp

La empresa Prototyp mantendrá un seguimiento por encima de mi proyecto. Haré una presentación del proyecto cuándo esté en una fase bastante avanzada en inglés, pues es el idioma que se utiliza en la empresa, al ser todos los integrantes de distintos países. Además, cada semana aproximadamente tendré una revisión de mi código por un programador senior[8] de la empresa, de tal modo que mejore mis habilidades escribiendo código.

1.3.3. Director de la empresa

Si bien antes he puesto a la empresa, me gustaría definir aparte al director de la empresa, Pontus Österberg, que será quien vaya siguiendo parte del proyecto a través de reuniones semanales. Es uno de los principales interesados en la aplicación y quién aceptará/rechazará (dependiendo del resultado) las funcionalidades que introduzca en el sistema a través de un gestor de proyecto usado en la empresa.

1.3.4. Donantes

Son los principales usuarios de la aplicación. Son las personas que tienen interés en apoyar proyectos medioambientales, además de querer proyectar sus ganas de mejorar el mundo a través de un regalo a otra persona (receptor).

1.3.5. Receptores

Los receptores son las personas que reciben regalos de parte de donantes. Éstas personas reciben la experiencia que ofrece la aplicación web.

1.3.6. Administradores del sistema

Los administradores del sistema se encargan de verificar que los proyectos de caridad situados en la aplicación web están en actualizados, además de monitorizar las donaciones y los envíos a los receptores.

[8] Programador que puede desarrollar funcionalidades complejas y ejecutar proyectos de gran envergadura.

1.3.7. ONGs de medio ambiente

Son las empresas sin ánimo de lucro que impulsan proyectos de mejora para el medio ambiente y que están abiertos a donaciones externas.

1.3.8. Correos

Los regalos para los donantes se han de enviar por algún medio. Por lo que es necesario una empresa que gestiona los envíos de cartas hacía los receptores. Al ser este un proyecto pequeño, los administradores del sistema usarán la empresa ‘Correos’ para enviar cartas.

1.3.9. Autor del TFG

Yo, como autor del TFG, me encargaré de gestionar planificar e implementar el proyecto. Además de redactar esta memoria que contendrá todo lo que tenga que ver con el TFG.

2. Alcance del proyecto

En todo proyecto se ha de establecer un límite respecto a las cosas que se quiere alcanzar. Un alcance bajo haría que nuestra aplicación tuviese poco valor, sobretodo hoy en día que con la llegada de Internet, hay miles y miles de aplicaciones que resuelven todo tipos de problemas por doquier. Sin embargo, un alcance alto supondría un trabajo demasiado alto como para ejecutarlo un solo estudiante, pues recordemos que ejerceré todos los roles en la construcción de este software. Dicho esto y habiendo introducido el problema que quiero solucionar en su apartado correspondiente, es hora de definir los objetivos principales que tengo en mente alcanzar.

2.1. Objetivos principales

Regalar gadgets[9] innecesarios es perjudicar al medio ambiente y cargar de cosas al destinatario de estos regalos. Por eso, el objetivo principal del proyecto es hacer que donar hacia proyectos sin ánimo de lucro en nombre de otras personas sea divertido, interactivo y conscience a otras personas. De este modo, agregando una experiencia digital, se intentará convertir un regalo encomiable a uno excelente.

Además, contribuir en la mejora del medio ambiente es otro punto a alcanzar. Es un problema que nos toca a todos pero poco eco se hace de ésto. Obviamente hay miles de tipos de problemas humanitarios que también se podría intentar ayudar introduciendolos en el sistema, pero según la filosofía de la empresa donde voy a realizar el TFG: “Es mejor empezar con algo pequeño y después expandirlo”.

2.2. Aspectos básicos

Dichas las idea generales que quiero alcanzar, a continuación voy a detallar los puntos claves que cubrirá mi proyecto y la manera en cómo se alcanzarán:

- Fácil de usar e intuitivo: Este es un aspecto esencial que ha de alcanzar la aplicación web. Al ofrecer un servicio donde hay dinero de por medio, se ha de hacer que los usuarios del sistema no tengan trabas al usar el servicio. O por ende acabarán buscando otra alternativa.

- Personalizable: El cliente ha de ser capaz de personalizar el regalo a través de un vídeo mensaje. De este modo, el regalo tiene un valor más personal.

- Variable: El dinero de las donaciones ha de ser totalmente variable. De manera que no se fuerce una cantidad al cliente.

- Aprendizaje interactivo: Se ha de intentar concienciar sobre problemas del medio ambiente a las personas que reciben los regalos de manera interactiva mediante un cuestionario.

[9] Un Gadget es un dispositivo que tiene un propósito y una función específica, generalmente de pequeñas proporciones.

- Experiencia digital: Los usuario han de ser capaces de regalar algo, pero ese “algo” ha de ofrecer una experiencia puramente digital. Es decir, no ha de ser ningún tipo de postal u objeto recordatorio.

- Contribución: Se ha de buscar una manera a partir de la cual se ayude a financiar proyectos que ayuden a mejorar el medio ambiente.

- Seguridad: Las operaciones que se realicen en el sistema que he de crear han de ser seguras. De tal manera que los usuario que utilicen el servicio que voy a ofrecer no se vean comprometidos.

2.3. Posibles obstáculos

En todo proyecto hay obstáculos y este no será una excepción. Para empezar, he de comentar que si bien he realizado alguna que otra aplicación web en asignaturas de la universidad, mis conocimientos son limitados y puede que encuentre retrasos en la planificación en mi proyecto debido a las tecnologías usadas.

Otro de los posibles obstáculos que podría encontrar es el deterioro de mi hardware. El ordenador en el que voy a realizar el proyecto tiene ya algunos años y eso se nota. No obstante, creo que podrá aguantar los siguientes meses y no tendré mucho problema.

Finalmente, he de comentar que mi aplicación tiene pensado utilizar servicios de terceros, con los cual, he de ‘fiarme’ en que estos no cambien bruscamente, que se mantengan estables los meses en los cuales estaré realizando el proyecto.

3. Metodología

En la empresa donde realizaré el TFG, se hace uso principalmente de la metodología Agile. Esta metodología ha estado muy de moda últimamente en el mundo del software y cada vez es más utilizada en startups[10]. Dejo por aquí una cita de una entrada de un blog que define esta metodología a la perfección:

"*'Agile' es mucho más que una metodología para el desarrollo de proyectos que precisan de rapidez y flexibilidad, es una filosofía que supone una forma distinta de trabajar y de organizarse. De tal forma que cada proyecto se 'trocea' en pequeñas partes que tienen que completarse y entregarse en pocas semanas. El objetivo es desarrollar productos y servicios de calidad que respondan a las necesidades de unos clientes cuyas prioridades cambian a una velocidad cada vez mayor.*" [2]

3.1. Scrum

Hay muchas implementaciones existentes para realizar proyectos ágiles, entre ellos Scrum, Extreme Programming, Crystal, Feature-Driven Development, etc [3].

La empresa donde haré el TFG ha escogido Scrum como método ágil, ya que en Scrum lo que se quiere es obtener resultados lo antes posibles, centrándose en el núcleo de la idea principal y modificando requisitos que a priori se saben que van a ser cambiantes. De hecho, Scrum es la implementación más famosa de Agile, tal y como dice el siguiente extracto:

"*El uso de los métodos ágiles continúa aumentando en tracción y Scrum es el método más ampliamente adoptado para el desarrollo de software ágil. Los niveles generales de interés en Scrum son por lo tanto considerables*" [4].

3.2. Herramientas de seguimiento

Pivotal Tracker: "*Es la herramienta de gestión de proyectos ágil elegida por desarrolladores de todo el mundo para la colaboración en tiempo real en torno a un backlog[11] priorizado*" [5]. Es la herramienta principal que se usa en mi empresa

PlanITpoker: Herramienta online que sirve para estimar proyectos ágiles. Se pondrán funcionalidades y se valorarán mediante un sistema de puntuación.

Git/Github: Git es una herramienta de control de versiones para el proyecto que realice. Todo el código que escriba estará subido en la web de Github[12].

[10] Startup es una empresa de nueva creación que comercializa productos y/o servicios con un modelo de negocio escalable el cual le permite un crecimiento rápido en el tiempo.
[11] El Backlog es una lista ordenada que contiene todo el trabajo pendiente de un proyecto.
[12] https://github.com/

Heroku: Es una plataforma que ofrece servicio de computación online. Cuando el proyecto alcance fases donde el software es estable, será subido a Heroku[13], de manera que cualquier persona pueda acceder a la aplicación web.

3.3. Proceso

Debido a que realizaré un proyecto ágil, lo primera que haré será definir historias de usuario que tendrá la aplicación, y una vez las tenga todas partiré el trabajo en iteraciones definidas previamente en la planificación temporal de mi proyecto, donde cada una de las iteraciones contendrá una cantidad de historias de usuario.

Una vez hecho esto, pasamos a la fase de desarrollo, donde la comunicación entre la empresa y el equipo de desarrollo va a ser clave. De hecho, después de cada iteración me reuniré con el director de la empresa para exponer los avances del proyecto. También he de comentar que gracias a la herramienta Pivotal Tracker, el director de la empresa podrá aceptar/rechazar las características que introduzca en el sistema, de tal modo que el proyecto se vea desde otro punto de vista y no sólo el mío.

También habrán reuniones semanales con un programador senior de la empresa que se encargará de revisar mi código con tal de que mis habilidades programando mejores y así desarrollar un software mejor, evitando el famoso código espagueti[14].

Finalmente, he de comentar que si bien la metodología usada es ágil, para la documentación de ésta memoria se escribirá es forma de metodología en cascada de tal modo que la memoria quedará más clara, ya que la especificación, diseño e implementación final se encuentran en sus correspondientes capítulos y no de forma dividida por iteraciones.

3.4. Métodos de validación

En este proyecto se hará uso de una herramienta de integración contínua llamada Travis CI[15]. Cada subida de código nueva en el repositorio Github, será comprobado por pruebas unitarias[16] que haya realizado yo. De manera que todo código nuevo será comprobado y estaré seguro de que no interferirá con código anterior.

[13] https://heroku.com/
[14] Código espagueti es un término negativo para los programas de computación que tienen una estructura de control de flujo compleja e incomprensible.
[15] https://travis-ci.com/
[16] Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código.

4. Planificación temporal

Después de estos años en la FIB he aprendido que se han de planificar las cosas para que salgan bien. Los imprevistos siempre están ahí, pero si se disponen de las herramientas necesarias para planificar y ejecutar las actividades de un proyecto de manera que el tiempo en llevarlo a cabo sea casi el esperado, ¿por qué no hacerlo?

4.1. Planificación general

Este apartado contendrá la descripción del proyecto desde un punto de vista general. Se indicarán los recursos necesarios para llevar a cabo el proyecto, el plan de acciones teniendo en cuenta las alternativas para las posibles desviaciones del proyecto y finalmente unas pequeñas consideraciones.

4.1.1. Planificación estimada del proyecto

El convenio con la empresa en la que hago el TFG inicia el día 4 de Febrero y finaliza el día 17 de Junio. Además, este acuerdo establecido entre la universidad y la empresa indica que mi carga de trabajo será de 735 horas. Aunque he de tener en cuenta que la empresa me comentó que tendría que participar un poco en otro proyecto aparte donde se espera que se cobre a un cliente por mi trabajo. Esta participación tendría lugar los días finales al contrato. Por lo que mi planificación se basará en 600 horas y las 135 horas restante las tendré para contratiempos y para la participación en el proyecto externo que me indicará la empresa, por lo que este TFG tendrá una duración estimada de 4 meses aproximadamente a jornada completa.

4.1.2. Recursos

Los recursos que se utilizarán en este proyecto son:

Recursos humanos:

	Finalidad	Horas semanales
Yo	Ejercer la mayoría de los roles que hay en el desarrollo de un software	40

Tabla 2. Recursos humanos.

Recursos materiales:

Recurso	Tipo	Finalidad
Ordenador portátil Toshiba Satellite p50, 8gb de RAM, 128gb ssd, Linux Ubuntu	Equipo	Desarrollo de la aplicación web y redacción de la memoria

RubyMine 2018.3.5	Herramienta de desarrollo	Desarrollo de la aplicación web (back-end[17])
WebStorm 2018.3.5	Herramienta de desarrollo	Desarrollo de la aplicación web (front-end[18])
Google suite	Herramienta de desarrollo	Redacción de la memoria y seguimiento del trabajo en hojas de cálculo y almacenaje de archivos
Correo electrónico - Gmail	Herramienta de comunicación	Comunicación con la empresa y registro en servicios web
Correo electrónico - FIB	Herramienta de comunicación	Comunicación con mi ponente / Curso GEP[19]
Pivotal Tracker	Herramienta de gestión	Planificación del proyecto
PlanITpoker	Herramienta de gestión	Calcular dificultad de funcionalidades del proyecto
Git	Herramienta de control	Controlar las versiones del código del proyecto
Github	Herramienta de control	Controlar las versiones del código del proyecto de forma remota
Restlet Client	Herramienta de desarrollo	Probar el back-end del proyecto
Heroku	Herramienta de desarrollo	Poder publicar la aplicación en un servidor de forma que sea accesible a través de una URL pública
Travis CI	Herramienta de desarrollo	Herramienta que me permite comprobar que el código nuevo que genere no interfiera con el anterior, además de ofrecer herramientas de automatización

Tabla 3. Recursos materiales.

4.1.3. Plan de acción y valoración de alternativas

Durante en transcurso del desarrollo de este TFG, pueden ocurrir diversas desviaciones, por lo que es esencial tener una alternativa para cada una de éstas en caso de que ocurriera.

[17] "*Cuando hablamos del 'front-end' de una web, de lo que realmente estamos hablando es de la parte de la web que puedes ver e interactuar.*" [6]
[18] El back-end es la parte de la lógica que hay en una aplicación para que funcione.
[19] Sigla correspondiente a: Gestión De Proyectos

Desviación por una mala planificación:
Si la planificación es errónea y me sobra tiempo, se intentará poner funcionalidades extra a la aplicación o a insertar aquellas que fueron descartadas para centrarme solo en el núcleo del proyecto. Además de dedicar más tiempo a partes no funcionales del proyecto (como la seguridad, testing, etc.).

Sin embargo si me falta tiempo, se hará uso de una parte de las 135 horas que tengo guardadas para contratiempos y la participación en un proyecto externo de la empresa, para una iteración adicional. Además, si aún así preveo que me quedaré muy corto de tiempo, me centraré solo en las funcionalidades esenciales del proyecto, dejando de lado funcionalidades secundarias.

Desviación por cambios en los servicios de terceros:
En la aplicación web que tengo pensando realizar es posible que servicios de terceros dejen de ofrecer soporte a desarrolladores o que incluso salgan nuevas versiones dejando las antiguas de lado o aún peor, obsoletas. Sin embargo, si eso pasa intentaré buscar alternativas a esos servicios. En principio los servicios que utilizaré no serán de pago, pero si se diese el caso, la empresa me ha comentado que podría contribuir con una cantidad monetaria.

Desviación causada por fallo en el ordenador:
El ordenador en el que trabajo para hacer el proyecto tiene ya algunos años encima. Es por eso que si dejase de funcionar o se estropea, pediré un portátil a la empresa o compraré un portátil con el dinero del convenio.

4.2. Descripción de las tareas

4.2.1. Puesta en marcha

En esta fase inicial del proyecto, me dediqué principalmente a configurar mi entorno de trabajo y a planear las distintas funcionalidades que tendrá la aplicación mediante el análisis de requisitos. Además de dedicar horas al aprendizaje de las diversas tecnologías que tengo planeado usar, de tal modo de que cuando empiece la fase de desarrollo realice un código de mejor calidad.

4.2.2. Gestión del proyecto

Esta fase del proyecto se centrará únicamente en la elaboración de los documentos entregables que se han de realizar para la asignatura obligatoria GEP. Esta asignatura se basa en cuatro entregables los cuales se van a ir realizando de forma secuencial, teniendo en cuenta las fechas de entrega que establece la asignatura, que en principio son fijas.

4.2.3. Desarrollo de la aplicación

Esta parte del proyecto se basará en el desarrollo de la aplicación web que tengo pensado hacer. La dependencia que tenemos son las anteriores fases, pues antes de desarrollar he de planear el proyecto. El desarrollo se verá dividido en diversas iteraciones donde cada una de ellas será dependiente de la anterior, pues se hará todo de forma secuencial. Estas iteraciones tendrán a su vez las siguientes fases:

- Diseño del front-end: Realizar el diseño de las vistas de la aplicación.
- Diseño de la base de datos: Ir expandiendo el diseño de la base de datos para que sea acorde a las funcionalidades que se va a ofrecer.
- Implementación del código: El proceso de escribir código en el proyecto.
- Pruebas unitarias: Las diferentes pruebas para las distintas funcionalidades de la aplicación.
- Despliegue: El proceso de subir el código estable a producción[20].
- Documentación: Escritura de la memoria, documentar el proyecto en sí.

4.2.4. Documentación y presentación

Finalmente esta será la fase donde terminaré toda la documentación, revisando errores y reestructurando la memoria del TFG (que se irá haciendo durante todo el proyecto).

Además, también habré de dedicar tiempo a preparar la defensa del TFG, es decir, realizar la presentación y ensayar.

4.3. Calendario

4.3.1. Estimación de horas

En esta tabla están las estimaciones de horas para cada una de las tareas del proyecto. Debido a que GEP se imparte durante las semanas que tengo planteado empezar a trabajar en el proyecto, las primeras iteraciones contienen menos horas y es por ello que el desarrollo aparece dos veces en la tabla.

Tarea	Responsable	Horas
Puesta en marcha		80
Configuración entorno	Programador	10
Aprendizaje tecnologías	Programador	40
Definir requisitos	Analista	30
Gestión del proyecto		75
Entrega 1: Alcance y contextualización	Jefe de proyecto	25
Entrega 2: Planificación temporal	Jefe de proyecto	11
Entrega 3: Gestión económica y sostenibilidad	Jefe de proyecto	13

[20] Entorno preparado para ser usado por usuarios finales.

Entrega 4: Presentación oral y documento final	Jefe de proyecto	26
Desarrollo de la aplicación 1		55 x 3 iteraciones = 165
Diseño web	Diseñador	8
Diseño base de datos	Administrador de base de datos	8
Implementación del código	Programador	20
Pruebas unitarias	Tester	8
Despliegue	Jefe de proyecto	2
Documentación	Jefe de proyecto	9
Desarrollo de la aplicación 2		80 x 3 iteraciones = 240
Diseño web	Diseñador	10
Diseño base de datos	Administrador de base de datos	10
Implementación del código	Programador	30
Pruebas unitarias	Tester	14
Despliegue	Jefe de proyecto	2
Documentación	Jefe de proyecto	14
Documentación y presentación		40
Redacción memoria	Jefe de proyecto	20
Presentación oral	Jefe de proyecto	20
Horas totales		**600**

Tabla 4. Estimaciones de horas.

4.3.2. Diagrama de Gantt

He aquí el diagrama de Gantt obtenido:

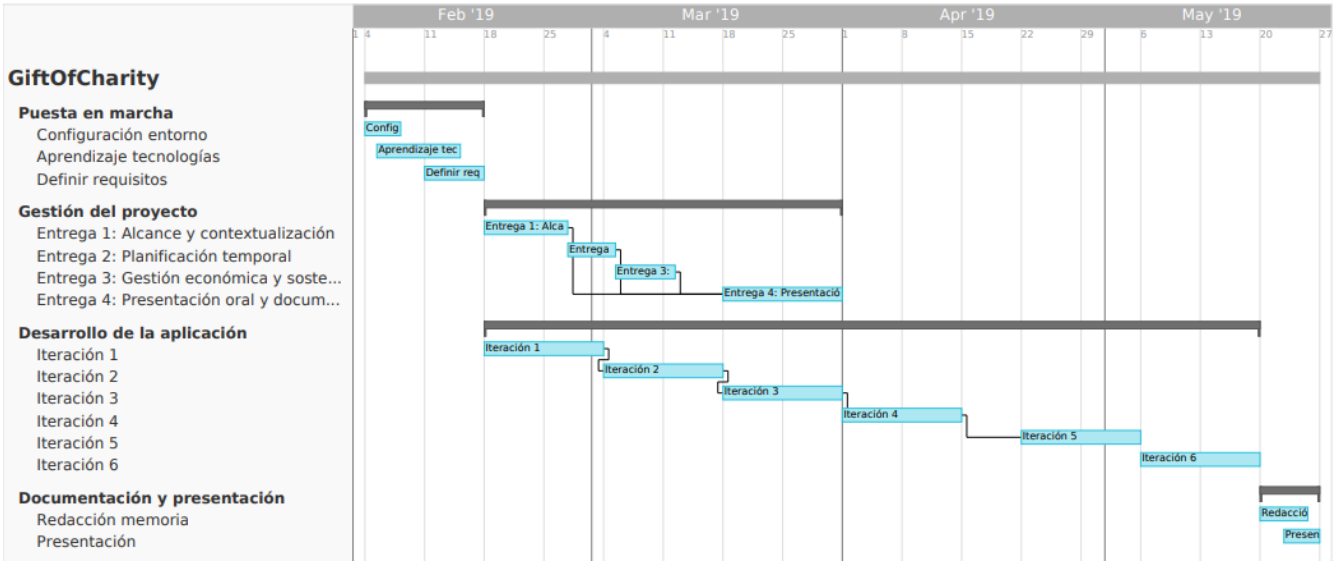


Imagen 1. Diagrama de Gantt.

5. Gestión económica

Una vez visto el apartado de planificación vamos a analizar los costes del proyecto, por lo que realizaré un desglose de los diversos costes que tendrá el desarrollo del proyecto a partir de la planificación.

5.1. Costes recursos humanos

En la tabla 5 vemos los costes de los recursos humanos, donde hay diversos roles el respectivo coste según la publicación del estudio de remunerabilidad de Michael Page del año 2018 [8], cogiendo los sueldos mínimos anuales y dividiéndolo en 12 pagas a jornada completa.

Rol	Horas estimadas	Remuneración (€/hora)	Coste (€)
Jefe de proyecto	196	21	4116,00
Programador	200	9	1800
Analista	30	13	390
Diseñador	54	9	486
Administrador de bases de datos	54	13	702
Tester	66	9	594
Total	**600**		**8088,00**

Tabla 5. Costes recursos humanos.

5.2. Costes directos por actividad

A modo de mirar los costes desde otro punto de vista, en la siguiente tabla 6 haré un desglose de los costes por actividad teniendo en cuenta las remuneraciones usadas antes.

Actividad	Recurso	Horas estimadas	Remuneración (€/hora)	Coste (€)
Puesta en marcha		80		840,00
Configuración entorno	Programador	10	9	90
Aprendizaje tecnologías	Programador	40	9	360
Definir requisitos	Analista	30	13	390
Gestión del proyecto		75		1575
Entrega 1: Alcance y contextualización	Jefe de proyecto	25	21	525
Entrega 2: Planificación temporal	Jefe de proyecto	11	21	231
Entrega 3: Gestión económica y sostenibilidad	Jefe de proyecto	13	21	273

Entrega 4: Presentación oral y documento final	Jefe de proyecto	26	21	546
Desarrollo de la aplicación		405		4833
Diseño web	Diseñador	54	9	486
Diseño base de datos	Administrador de base de datos	54	13	702
Implementación del código	Programador	150	9	1350
Pruebas unitarias	Tester	66	9	594
Despliegue	Jefe de proyecto	12	21	252
Documentación	Jefe de proyecto	69	21	1449
Documentación y presentación		40		840
Redacción memoria	Jefe de proyecto	20	21	420
Presentación oral	Jefe de proyecto	20	21	420
Total		**600**		**8088**

Tabla 6. Costes directos por actividad.

5.3. Costes indirectos

En la siguiente tabla están los costes indirectos del proyecto. Cabe destacar que los gastos básicos de la oficina (internet, luz, agua) no se tendrán en cuenta pues no se ha introducido ningún cambio ni variación en el consumo después de mi llegada a la empresa, estos siguen funcionando igual desde hace meses. Además, todo software y servicio usado en el proyecto es de libre uso y de los que no, utilizo los planes gratuitos que hay.

Producto	Unidades	Precio unitario	Coste (€)
Oficina	4 meses	0 € / mes	0,00
Transporte (T-jove)	1 tarjeta	142,00 €[21]	142,00
Transporte (T-10 2 zonas)	4 tarjetas	20,10 €[1]	80,40
Amortización portátil	4 meses	0,12 € / hora[22]	72,00
Impresiones en papel	800 páginas[23]	0,04 € / página	32,00
Software y servicios	4 meses	0 € / mes	0,00
Total			**326,40**

[21] Precios extraídos de la web de Transports Metropolitans de Barcelona [9].
[22] Resultado extraído de la fórmula: 1000€ precio inicial del portátil / (4 años de vida útil * 251 días laborables * 8 horas diarias).
[23] Supongamos 200 páginas aproximadamente de la memoria acabada y se tendrán que hacer 4 copias (3 miembros del tribunal más el ponente del TFG).

Tabla 7. Costes indirectos.

5.4. Contingencias

Respecto a las contingencias, reservamos un porcentaje del 15% tanto en los costes directos como indirectos con tal de cubrir gastos que no se han podido planificar en este documento.

Producto	Porcentaje	Precio (€)	Coste (€)
Costes directos	15%	8088,00	1213,2
Costes indirectos	15%	326,40	48,96
Total		**8414,40**	**1262,16**

Tabla 8. Contingencias.

5.5. Imprevistos

En los imprevistos nombrados en el apartado de valoración de alternativas, tenemos el retraso del proyecto, donde su precio es de 110€, ya que se suma el precio de una iteración más el de transporte y uso del portátil con una probabilidad del 35%. Además, también tengo en cuenta si el portátil se estropea, con una probabilidad del 10%.

Imprevisto	Probabilidad	Precio (€)	Coste (€)
Retraso del proyecto	35%	110	39€
Portátil estropeado	10%	600	60€
Total		**710**	**99€**

Tabla 9. Imprevistos.

5.6. Presupuesto

Finalmente, en la siguiente tabla 10 se muestra el coste total del proyecto.

Concepto	Coste (€)
Costes directos	8088,00
Costes indirectos	326,40
Contingencia	1262,16
Imprevistos	98,50
Total	**9775,06**

Tabla 10. Presupuesto final.

5.7. Control de gestión

Con tal de controlar las diferentes partes del proyecto y prevenir posibles desviaciones, se tendrán en cuenta: horas estimadas y reales, precio/hora estimadas y reales también, así como las diferencias

entre las estimaciones y lo real. Estas variables tienen la finalidad de obtener las desviaciones en precio: (costes estimados - costes reales) * horas reales y las desviaciones en consumo: (horas estimadas - horas reales) * costes estimados, de tal forma que se verá reflejado en el presupuesto final.

5.8. Reflexión

Finalmente, he de decir que este proyecto es sin ánimo de lucro, por lo que no tengo en cuenta ningún tipo de beneficio recibido a partir de las donaciones, ya que la idea es que todo el dinero vaya a parar a proyectos para la mejora del medio ambiente. Además, respecto a los regalos de los receptores, no se ampliará el presupuesto con esto, pues la carta que se envía al supuesto receptor se puede simular perfectamente y aunque sería una cantidad muy pequeña de dinero, no se tiene en mente aumentar más el presupuesto.

Este presupuesto está realizado en base a los costes que conlleva implementar el proyecto, pero respecto a la vida útil del proyecto, se ha de comentar que habrá de existir un administrador que se encargue de realizar la entrega de cartas, así como el mantenimiento general del sistema de tal manera de que si hubiese algún problema con la web, se pudiese solucionar en un corto periodo de tiempo.

6. Requisitos del sistema

En Ingeniería del Software, los requisitos son necesidades documentadas sobre la funcionalidad, forma y contenido de un producto / servicio [10]. Con tal de que la aplicación web que vamos a construir sea útil y de calidad vamos a definir estos requisitos en dos partes. Por un lado tenemos los requisitos funcionales que hacen referencia a las funcionalidades de la aplicación que estarán en forma de historias de usuario y por otro lado tenemos los requisitos no funcionales en forma de tabla.

6.1. Historias de usuario

Las historias de usuario son descripciones cortas y esquemáticas que sirven principalmente para resumir una necesidad del usuario al utilizar un producto / servicio. Ésta forma de resumir necesidades se ha hecho muy famosa debido a la metodología Agile (que es la que se usa en este proyecto), con lo cual en este apartado vamos a contener todas las historias de usuario que se encuentran dentro del alcance del proyecto organizadas en grupos llamadas épicas.

6.1.1. Proyectos de caridad

En esta épica se centrarán las historias de usuario que tengan que ver con el producto que se vende en la web, que en este caso son los proyectos de caridad. Los administradores han de administrar estos productos, pudiendo añadirlos, borrarlos o editarlos. Además, los visitantes han de ser capaces de ver estos productos en alguna vista para tener más información sobre éstos y así poder comprarlos.

Listado de proyectos de caridad: Como visitante de la web quiero poder ver una lista de proyectos de caridad disponibles para comprar en la vista principal de la aplicación.

Información de un proyecto de caridad: Como visitante de la web quiero poder ver información de un proyecto de caridad en específico tal como su descripción, imagen, etc. en una vista para saber más información del proyecto.

Creación de un proyecto de caridad (adm): Como administrador quiero poder crear proyectos de caridad para introducirlos en la aplicación y que los usuarios puedan verlos disponibles en la vista principal.

Eliminación de un proyecto de caridad (adm): Como administrador quiero poder eliminar proyectos de caridad para quitarlos de la aplicación.

Edición de un proyecto de caridad (adm): Como administrador quiero poder editar un proyecto de caridad para actualizarlo con información correcta como por ejemplo cambiar su descripción.

Listado de proyectos de caridad (adm): Como administrador quiero poder ver una lista de todos los proyectos de caridad introducidos en el sistema con información básica como su nombre, descripción, fecha de creación, etc. para tener una visión general sobre éstos y además tener la opción de desencadenar acciones como la eliminación de proyectos seleccionando varios elementos de la lista.

Habilitar/Deshabilitar proyecto de caridad (adm): Como administrador quiero poder habilitar o deshabilitar un proyecto de caridad para tener control sobre los proyectos que quiero que los usuarios vean disponibles en la aplicación web.

6.1.2. Donaciones

Ésta épica contendrá todas las historias de usuario que tengan que ver con las donaciones. Los donantes han de ser capaces de comprar proyectos de caridad en distintos pasos: Subir vídeo mensaje, rellenar datos de envío, rellenar datos del donante, realizar la compra con un método de pago. Y no nos olvidemos de los administradores han de tener operaciones básicas para saber toda la información de las donaciones del sistema.

Rellenar formulario (receptor): Como donante quiero poder completar un formulario con datos de envío del receptor como su población, código postal, dirección, etc. para que el regalo llegue a su destino.

Rellenar formulario (donante): Como donante quiero poder completar un formulario con datos míos como mi e-mail para dejar constancia de mi donación. En caso de problemas tengo la seguridad de que hay una relación entre la compra y yo.

Subir vídeo: Como donante quiero poder subir un video mensaje en el proceso de compra de un proyecto de caridad para que el receptor pueda verlo. Una vez el vídeo está subido se me mostrará una previsualización para asegurarme de que el vídeo subido es el correcto.

Demo de una donación: Como donante quiero poder ver una demo de la donación para asegurarme que me gusta, ésta demo me llevará a una web que mostrará exactamente lo mismo que verá el receptor.

Realizar donación: Como donante quiero poder pagar una donación con un método de pago conocido para poder completar la donación. He de ser consciente del estado de mi compra a través de mensajes informativos en la vista en la que me encuentre.

Recibo de compra: Como donante quiero recibir un e-mail con información sobre mi compra para tener la seguridad de que he realizado la compra correctamente y tener algo con lo que reclamar en caso de que hubiese algún problema.

Listado de donaciones (adm): Como administrador quiero poder ver una lista de todas las donaciones realizadas en la tienda para ver información básicas de éstas como la fecha de realización, estado actual, etc., además de poder acceder a una vista específica para cada donación.

Información de una donación (adm): Como administrador quiero poder ver la información de una donación en una vista específica para ver toda la información que el sistema posee.

Editar donación (adm): Como administrador quiero poder editar la información de una donación en una vista específica para poder modificar cambios incorrectos que se hayan podido producir en el proceso de donación por parte del donante.

6.1.3. Regalos

En ésta épica se situarán las historias de usuario que tengan que ver con los regalos que reciben los receptores. Éstos han de ser capaces de abrir su regalo generado a través de la donación siguiendo los pasos que indica la carta física que reciben. Podrán visualizar un vídeo mensaje de parte del donante (que es la persona que lo envía) en la web a la que hace referencia la carta. A su vez, los administradores podrán realizar pequeñas acciones como marcar un regalo como enviado en el apartado de donaciones.

Acceso al regalo: Como receptor quiero poder acceder a la web a la que hace referencia la carta que he recibido para que pueda ver mi regalo a través de un método de acceso no definido aún. Éste método puede ser un código QR o un link que te lleve directamente al regalo o una contraseña para introducir en algún apartado de la aplicación web.

Consumir un vídeo mensaje: Como receptor quiero poder consumir un mensaje en forma de video para ver lo que hay en él. Éste vídeo será el que un donante se haya encargado de subir para que yo lo reproduzca.

Notificación de regalo abierto: Como donante quiero poder recibir una notificación, en forma de e-mail, cuando el regalo enviado al receptor de la donación ha sido abierto. Esto me servirá para saber cuándo mi regalo ha sido recibido.

Marcar regalo como enviado (adm): Como administrador quiero poder marcar un regalo como enviado para que los donantes sepan que su regalo ha sido enviada al receptor. Ésto lo podré hacer en el panel de administrador de una forma intuitiva.

Descargar carta (adm): Como administrador quiero poder descargar la carta que recibirán los receptores en formato PDF para que pueda enviarlo por correo.

6.1.4. Cuestionarios

Listado de cuestionarios (adm): Como administrador quiero poder ver una lista de todos los cuestionarios introducidos en el sistema con información básica como número de preguntas, título, etc. para tener una visión general sobre éstos.

Crear cuestionario (adm): Como administrador quiero poder crear cuestionarios para introducirlos en la aplicación y pueda ser enlazado a cualquier proyecto de caridad.

Eliminar cuestionario (adm): Como administrador quiero poder eliminar cuestionarios para que no se pueda enlazar a proyectos de caridad.

Editar cuestionario (adm): Como administrador quiero poder editar un cuestionario para actualizar las preguntas y respuestas que tiene.

Consumir cuestionario: Como receptor quiero poder consumir un cuestionario con diversas preguntas para concienciarme sobre el tema del cuestionario.

6.1.5. Otras tareas

Finalmente, en ésta épica tendremos todas las historias de usuario que vayan saliendo y que no correspondan a ninguna de las otras épicas.

Información estática sobre la web: Como visitante quiero leer información general sobre la web para saber más de ésta, tales como la finalidad, cómo funciona, contacto, etc.

Invitación al panel de administración: Como administrador quiero poder añadir a personas al panel de administración a través de una invitación para que pueda empezar a administrar la aplicación.

6.2. Requisitos no funcionales

En este apartado pondremos todos los requisitos que no tienen que ver con las funcionalidades que tiene el sistema. Es decir, estamos hablando de características de funcionamiento y de hecho es por eso que también suelen llamarse atributos de calidad. Para definir estos requisitos, se van a dividir en grupos según el tipo descrito por el documento Volere [11], que es una plantilla que se usa como base para definir requisitos. De este documento, he escogido los que los requisitos no funcionales que más me han llamado la atención y los que considero más importantes para este proyecto.

Debido a que muchos de estos requisitos no pueden ser valorados cuantitativamente, el criterio de satisfacción de algunos de éstos se basará en preguntas a personas tanto de la empresa como fuera de ella.

6.2.1. Requisitos de percepción

Aquí situaremos los requisitos que tengan que ver con las percepción del usuario final. Estamos hablando de una tienda online y la percepción del usuario es muy importante si queremos conseguir la finalidad principal que es que los productos que se tienen en la tienda sean comprados.

Tipo (Volere)	10a. Apariencia
Descripción	La apariencia general de la aplicación web ha de ser atractiva.
Justificación	Un sistema atractivo hace que los usuarios quieran usarlo. En este caso es muy importante que se quiera usar pues estamos hablando de una aplicación web que busca que se realicen donaciones
Criterio de satisfacción	Al 80% de los usuarios preguntados les ha de parecer atractivo.

Tabla 11. Requisito no funcional - Apariencia.

Tipo (Volere)	10b. Diseño
Descripción	El diseño del sistema sigue los estándares actuales.
Justificación	Si los usuarios del sistema ven que la aplicación web se parece a las demás, se sentirá cómodo y lo utilizará.

Criterio de satisfacción	Al 90% de los usuarios preguntados les ha de parecer moderno.

Tabla 12. Requisito no funcional - Diseño.

6.2.2. Requisitos de usabilidad

Otro punto importante es la usabilidad del sistema. Los usuarios han de sentirse como si estuvieran en su zona de confort, como si estuvieran accediendo a su gestor de mensajes de toda la vida. Es por ello que también nos centraremos en la usabilidad de la aplicación web, que ha de ser fácil de usar y sencillo de aprender con tal de que puedan usarlo personas de todas las edades sin complicación alguna.

Tipo (Volere)	11a. Facilidad de uso
Descripción	El sistema ha de ser fácil de utilizar.
Justificación	Cuánto más fácil sea de utilizar la aplicación web, más probabilidades hay de que un usuario realice una donación.
Criterio de satisfacción	Al 80% de los usuarios preguntados les ha de fácil de usar.

Tabla 13. Requisito no funcional - Facilidad de uso.

6.2.3. Requisitos de rendimiento

En los requisitos de rendimiento del sistema no solo estamos hablando de la velocidad de respuesta que tiene el servidor. En este grupo englobamos varios requisitos que tienen que ver con el rendimiento general del sistema, ya sea la disponibilidad, tolerancia a fallos, entre otros.

Tipo (Volere)	12a. Velocidad
Descripción	Las respuestas del sistema han de ser entregadas al usuario en un corto periodo de tiempo.
Justificación	Si el sistema es lento, los usuarios (posibles donantes) pierden interés en usarlos.
Criterio de satisfacción	Habrás *tests* que pasarán dependiendo del tiempo de respuesta del servidor. Si es menor a un límite de tiempo establecido será correcto, si en cambio es mayor, será incorrecto. Han de pasar todos los tests.

Tabla 14. Requisito no funcional - Velocidad.

Tipo (Volere)	12d. Fiabilidad y disponibilidad
Descripción	El sistema debe de estar disponible la mayor parte del tiempo.
Justificación	Los usuarios han de ser capaces de acceder al sistema a prácticamente cualquier hora del día.
Criterio de satisfacción	El sistema no puede estar caído más de 5 horas a la semana.

Tabla 15. Requisito no funcional - Fiabilidad y disponibilidad.

Tipo (Volere)	12f. Capacidad
Descripción	El sistema almacenará la menor cantidad de información posible de los usuarios.
Justificación	Cuanto menor información se posea de los usuarios mejor. De esta manera el consumo de memoria del sistema será reducido y por ende tendrá menos trabajo y mayor rendimiento en general.
Criterio de satisfacción	Los vídeos subidos por parte de los donantes han de tener un tamaño máximo de 200 mb.

Tabla 16. Requisito no funcional - Capacidad.

6.2.4. Requisitos de mantenimiento

En este grupo estarán los requisitos no funcionales que tengan que ver con el mantenimiento del código existente que constituye el sistema.

Tipo (Volere)	14a. Mantenimiento
Descripción	El código nuevo que vaya surgiendo después de cada *sprint* no ha de interferir con código ya existente.
Justificación	No se pueden ir acumulando errores. Se ha de pensar a largo plazo para que el sistema tenga éxito.

Criterio de satisfacción	Se ha de utilizar una herramienta de integración continua, de manera que los *tests* se vayan probando cada vez que se suba código nuevo al gestor de código usado (*Github*).

Tabla 17. Requisito no funcional - Mantenimiento.

6.2.5. Requisitos de seguridad

La seguridad en una aplicación online donde se realizan donaciones es muy importante, ya que estamos hablando del dinero de los usuarios. Las transacciones han de ser seguras y los datos de éstas han de permanecer visibles sólo a los administradores del sistema. De ninguna manera se ha de permitir el acceso de usuarios normales a información confidencial o a la posibilidad de que realicen acciones que intervengan con el flujo de la aplicación.

Tipo (Volere)	15a. Acceso
Descripción	El sistema debe gestionar los accesos al panel de administrador y asegurarse de que sólo los administradores pueden acceder.
Justificación	Desde el panel de administrador se pueden realizar operaciones que pueden afectar a los usuarios del sistema de forma negativa si se hace un uso incorrecto.
Criterio de satisfacción	Solo los usuarios administradores pueden acceder al panel de administrador a través de un login.

Tabla 18. Requisito no funcional - Acceso.

6.2.6. Requisitos de cumplimiento legal

Finalmente, hemos de tener en cuenta las leyes actuales que afectan al proyecto. El proyecto no tendrá en cuenta todas las leyes digitales existentes, pero sí que tendrá en cuenta las más conocidas.

Tipo (Volere)	17a. Legal
Descripción	Un sistema serio y actual ha de estar al tanto de las leyes actuales.
Justificación	El sistema ha de cumplir las leyes en las que se vea implicado.

Criterio de satisfacción	La aplicación web mostrará un aviso para cookies si se usan y mantendrá las contraseñas de los usuarios encriptadas. Además de permitir borrar la información de los usuarios según dice el reglamento general de protección de datos [12].

Tabla 19. Requisito no funcional - Legal.

7. Especificación

En este apartado se situará la especificación del sistema. La especificación es independiente de la tecnología y responde a la siguiente pregunta: ¿qué hace el sistema?

Primero de todo se especificarán detalladamente las historias de usuario definidas en el capítulo anterior y finalmente se pasará al esquema conceptual de los datos.

7.1. Especificación de las historias de usuario

Si bien las historias de usuario son útiles, a veces son demasiado simples y hacen falta más detalles. Es por ello que voy a detallar cada una de las historias de usuario con unos criterios de aceptación, que no son más que reglas definidas que han de ser respetadas para que las historias de usuario se consideren cumplidas [13], y unos escenarios que describen de forma genérica las situaciones que pueden ocurrir en la aplicación.

7.1.1. Proyectos de caridad

Historia de usuario	Listado de proyectos de caridad
Tarjeta	**Como** visitante de la web **quiero** poder ver una lista de proyectos de caridad disponibles **para** comprar en la vista principal de la aplicación.
Criterios de aceptación	- El usuario debe ser capaz de ver al menos un proyecto de caridad disponible. - El usuario debe poder seleccionar un proyecto de caridad y redirigir al usuario a otra vista con información del proyecto.
Escenarios	**Escenario 1**: Visualizar proyectos de caridad Dado un usuario de la aplicación, cuando entra en la vista principal del sitio web entonces debe poder visualizar una lista de proyectos de caridad a los que acceder. **Escenario 2**: Acceder a un proyecto de caridad Dado un usuario de la aplicación, cuando selecciona un proyecto de caridad, la aplicación le redirige a una vista dónde se encuentra información del proyecto.

Tabla 20. Especificación historia de usuario - Listado de proyectos de caridad.

Historia de usuario	Información de un proyecto de caridad

Tarjeta	**Como** visitante de la web **quiero** poder ver información de un proyecto de caridad en específico tal como su descripción, imagen, etc. en una vista **para** saber más información del proyecto.
Criterios de aceptación	- El usuario debe ser capaz de ver información correcta y actualizada del proyecto de caridad. - La aplicación ha de mostrar un mensaje cuando se intenta acceder a un proyecto de caridad inexistente. - El sistema ha de devolver información correcta y actualizada. - El usuario debe poder seleccionar el proyecto de caridad y redirigir al usuario a otra vista donde puede empezar el proceso de donación.
Escenarios	**Escenario 1**: Visualización de un proyecto de caridad existente Dado un usuario de la aplicación, cuando accede a la vista de la visualización de un proyecto de caridad existente, la aplicación le muestra información varia sobre el proyecto de caridad al usuario. **Escenario 2**: Visualización de un proyecto de caridad inexistente Dado un usuario de la aplicación, cuando accede a la vista de la visualización de un proyecto de caridad inexistente, la aplicación de devuelve un mensaje de error. **Escenario 3**: Empezar proceso de donación Dado un usuario de la aplicación, cuando decide empezar el proceso de donación, la aplicación le redirige a una vista donde hacerlo.

Tabla 21. Especificación historia de usuario - Información de un proyecto de caridad.

Historia de usuario	Creación de un proyecto de caridad (adm)
Tarjeta	**Como** administrador **quiero** poder crear proyectos de caridad **para** introducirlos en la aplicación y que los usuarios puedan verlos disponibles en la vista principal.
Criterios de aceptación	- El usuario debe ser capaz de introducir datos en un formulario. - El sistema debe verificar que los datos sean correctos. - El sistema ha de ser capaz de registrar el proyecto de caridad. - El usuario ha de ver un comunicado sobre el estado de la creación del proyecto de caridad.
Escenarios	**Escenario 1**: Creación del proyecto de caridad con datos válidos Dado un administrador del sistema en el panel de administración, cuando decide crear un nuevo proyecto de caridad y rellena un

	formulario con datos correctos, la aplicación le confirma la creación del proyecto de caridad y le redirige a una vista con los datos de la creación del proyecto de caridad. **Escenario 2**: Creación del proyecto de caridad con datos inválidos Dado un administrador del sistema en el panel de administración, cuando decide crear un nuevo proyecto de caridad y rellena un formulario con datos incorrectos, la aplicación muestra un mensaje de error y le permite corregir los fallos.

Tabla 22. Especificación historia de usuario - Creación de un proyecto de caridad (adm).

Historia de usuario	Eliminación de un proyecto de caridad (adm)
Tarjeta	**Como** administrador **quiero** poder eliminar proyectos de caridad **para** quitarlos de la aplicación.
Criterios de aceptación	- El usuario debe ser capaz de eliminar un proyecto de caridad que no esté vinculado a ningún regalo. - El sistema debe verificar que el proyecto de caridad se pueda eliminar. - El usuario ha de ver un comunicado sobre el estado de la eliminación del proyecto de caridad.
Escenarios	**Escenario 1**: Eliminación de un proyecto de caridad no vinculado a ningún regalo Dado un administrador del sistema en el panel de administración, cuando decide eliminar un proyecto de caridad que no está vinculado a ningún regalo, la aplicación le redirige a la lista de proyectos de caridad y le muestra un mensaje que confirma la eliminación del proyecto. **Escenario 2**: Eliminación de un proyecto de caridad vinculado a al menos un regalo Dado un administrador del sistema en el panel de administración, cuando decide eliminar un proyecto de caridad que está vinculado a al menos un regalo, la aplicación le muestra un error.

Tabla 23. Especificación historia de usuario - Eliminación de un proyecto de caridad (adm).

Historia de usuario	Edición de un proyecto de caridad (adm)

Tarjeta	**Como** administrador **quiero** poder editar un proyecto de caridad **para** actualizarlo con información correcta como por ejemplo cambiar su descripción.
Criterios de aceptación	- El usuario debe ser de cambiar datos del proyecto de caridad. - El sistema debe verificar que los nuevos datos sean correctos. - El sistema ha de ser capaz de editar el proyecto de caridad. - El usuario ha de ver un comunicado sobre el estado de su edición al acabar.
Escenarios	**Escenario 1**: Edición del proyecto de caridad con datos válidos Dado un administrador del sistema en el panel de administración, cuando decide editar un proyecto de caridad con datos correctos, el sistema le muestra un mensaje de confirmación de la edición del proyecto. **Escenario 2**: Edición del proyecto de caridad con datos inválidos Dado un administrador del sistema en el panel de administración, cuando decide editar un proyecto de caridad con datos incorrectos, el sistema le muestra un mensaje de error.

Tabla 24. Especificación historia de usuario - Edición de un proyecto de caridad (adm).

Historia de usuario	Listado de proyectos de caridad (adm)
Tarjeta	**Como** administrador **quiero** poder ver una lista de todos los proyectos de caridad introducidos en el sistema con información básica como su nombre, descripción, fecha de creación, etc. **para** tener una visión general sobre éstos y además tener la opción de desencadenar acciones como la eliminación de proyectos seleccionando varios elementos de la lista.
Criterios de aceptación	- El usuario debe ser capaz de ver un lista de proyectos de caridad. - El usuario debe ser capaz de ver seleccionar proyectos de caridad para borrarlos. - El usuario debe ser capaz de seleccionar un proyecto de caridad de la lista y ser redirigido a una vista donde se encuentra toda la información del proyecto de caridad. - El sistema debe devolver todos los proyectos de caridad almacenados en la base de datos.
Escenarios	**Escenario 1**: Visualizar proyectos de caridad

	Dado un administrador del sistema en el panel de administración, cuando entra en la vista de proyectos de caridad, entonces debe poder visualizar una lista de proyectos de caridad con información muy básica. **Escenario 2**: Acceder a un proyecto de caridad Dado un administrador del sistema en el panel de administración, cuando selecciona un proyecto de caridad en la vista de proyectos de caridad, la aplicación le redirige a una vista dónde se encuentra toda la información del proyecto de caridad. **Escenario 3**: Selección de proyectos de caridad para borrarlos Dado un administrador del sistema en el panel de administración, cuando se encuentra en la vista de proyectos de caridad y selecciona varios, se le permite borrarlos y se eliminan del sistema si no están vinculados a ningún regalo.

Tabla 25. Especificación historia de usuario - Listado de proyectos de caridad (adm).

Historia de usuario	Habilitar/Deshabilitar proyecto de caridad (adm)
Tarjeta	**Como** administrador **quiero** poder habilitar o deshabilitar un proyecto de caridad **para** tener control sobre los proyectos que quiero que los usuarios vean disponibles en la aplicación web.
Criterios de aceptación	- El usuario debe ser capaz de habilitar o deshabilitar un proyecto de caridad. - El sistema debe actualizar correctamente el proyecto de caridad. - El sistema debe asegurarse de mostrar solo proyectos de caridad habilitados.
Escenarios	**Escenario 1**: Habilitar proyecto de caridad Dado un administrador del sistema en el panel de administración, cuando entra en la vista de proyectos de caridad y selecciona un proyecto de caridad, puede habilitarlo para que los usuarios que entren en la aplicación web puedan escogerlo para donar. **Escenario 2**: Deshabilitar proyecto de caridad Dado un administrador del sistema en el panel de administración, cuando entra en la vista de proyectos de caridad y selecciona un proyecto, puede deshabilitarlo para que los usuarios que entren en la aplicación web no puedan velo. Sin embargo, los receptores de regalos pueden acceder a la información del proyecto de caridad si

	está vinculado a su regalo.

Tabla 26. Especificación historia de usuario - Habilitar/Deshabilitar proyecto de caridad (adm).

7.1.2. Donaciones

Historia de usuario	Rellenar formulario (receptor)
Tarjeta	**Como** donante **quiero** poder completar un formulario con datos de envío del receptor como su población, código postal, dirección, etc. **para** que el regalo llegue a su destino.
Criterios de aceptación	- El usuario debe ser capaz de introducir los datos del receptor del regalo en un formulario. - El sistema debe verificar que los datos sean correctos. - El sistema ha de permitir seguir con el proceso de donación si los datos introducidos son correctos.
Escenarios	**Escenario 1**: Rellenar formulario del receptor con datos válidos Dado un donante, cuando decide rellenar datos válidos del receptor del regalo, la aplicación le permite seguir con el proceso de donación. **Escenario 2**: Rellenar formulario del receptor con datos inválidos Dado un donante, cuando decide rellenar datos inválidos del receptor del regalo, la aplicación resalta los errores y permite corregir los fallos.

Tabla 27. Especificación historia de usuario - Rellenar formulario (receptor).

Historia de usuario	Rellenar formulario (donante)
Tarjeta	**Como** donante **quiero** poder completar un formulario con datos míos como mi e-mail **para** dejar constancia de mi donación. En caso de problemas tengo la seguridad de que hay una relación entre la compra y yo.
Criterios de aceptación	- El usuario debe ser capaz de introducir sus datos en un formulario. - El sistema debe verificar que los datos sean correctos. - El sistema ha de permitir seguir con el proceso de donación si los datos introducidos son correctos.
Escenarios	**Escenario 1**: Rellenar formulario con datos válidos Dado un donante, cuando decide rellenar los datos del donante con información válida, la aplicación le permite seguir con el proceso de

	donación. **Escenario 2**: Rellenar formulario con datos inválidos Dado un donante, cuando decide rellenar los datos del donante con información inválida, la aplicación resalta los errores y permite corregir los fallos.

Tabla 28. Especificación historia de usuario - Rellenar formulario (donante).

Historia de usuario	Subir vídeo
Tarjeta	**Como** donante **quiero** poder subir un video mensaje en el proceso de compra de un proyecto de caridad **para** que el receptor pueda verlo. Una vez el vídeo está subido se me mostrará una previsualización para asegurarme de que el vídeo subido es el correcto.
Criterios de aceptación	- El usuario debe ser capaz de subir un vídeo al sistema. - El sistema debe verificar que el vídeo subido sea correcto. - El sistema ha de permitir seguir con el proceso de donación si el vídeo es correcto.
Escenarios	**Escenario 1**: Subir archivo válido Dado un donante, cuando decide subir un vídeo válido al sistema, la aplicación muestra una previsualización y permite seguir con el proceso de donación. **Escenario 2**: Subir archivo inválido Dado un donante, cuando decide subir un vídeo inválido al sistema (ya sea porque no es un vídeo o porqué el tamaño excede el límite establecido por el sistema), la aplicación muestra un mensaje de error y permite volver a subir un archivo al sistema.

Tabla 29. Especificación historia de usuario - Subir vídeo.

Historia de usuario	Demo de una donación
Tarjeta	**Como** donante **quiero** poder ver una demo de la donación **para** asegurarme que me gusta, ésta demo me llevará a una web que mostrará exactamente lo mismo que verá el receptor.
Criterios de aceptación	- El usuario debe ser capaz de ver una demo lo más realista posible.
Escenarios	**Escenario 1**: Visualización de demo Dado un donante, cuando decide ver una demo del regalo, podrá

	acceder a una vista que contendrá la demo, además de permitirle al donante seguir con el proceso de donación.

Tabla 30. Especificación historia de usuario - Demo de una donación.

Historia de usuario	Realizar donación
Tarjeta	**Como** donante **quiero** poder pagar una donación con un método de pago conocido **para** poder completar la donación. He de ser consciente del estado de mi compra a través de mensajes informativos en la vista en la que me encuentre.
Criterios de aceptación	- El usuario debe ser capaz de introducir la cantidad a donar en la vista. - El usuario debe ser capaz de pagar con un método de pago conocido. - El sistema debe verificar que la cantidad y el pago de la donación sea correcta. - El sistema ha de ser capaz de registrar la donación en el sistema. - El usuario ha de ver un comunicado sobre el estado de su donación.
Escenarios	**Escenario 1**: Cantidad incorrecta Dado un donante, cuando introduce una cantidad de incorrecta en el proceso de donación, el sistema ha de hacerle saber que se ha de colocar otra cantidad de dinero y esconde la opción de pagar. **Escenario 2**: Pago incorrecto Dado un donante, cuando intenta realizar un pago y el sistema lo rechaza, se le ha de mostrar un mensaje que le comunique el error y dar la posibilidad de volver a intentarlo. **Escenario 3**: Donación correcta Dado un donante, cuando intenta realizar una donación con una cantidad de dinero correcta y el servidor confirma la operación. Se le ha de mostrar un mensaje de confirmación y enviar un recibo de compra al donante.

Tabla 31. Especificación historia de usuario - Realizar donación.

Historia de usuario	Recibo de compra
Tarjeta	**Como** donante **quiero** recibir un e-mail con información sobre mi compra **para** tener la seguridad de que he realizado la compra

	correctamente y tener algo con lo que reclamar en caso de que hubiese algún problema.
Criterios de aceptación	- El usuario debe ser capaz de recibir un e-mail con información de la donación. - El sistema debe enviar un mail a la dirección del donante proporcionando datos de la donación.
Escenarios	**Escenario 1**: Envío de e-mail Dado un donante, cuando acaba de forma correcta el proceso de donación, el sistema le ha de enviar un e-mail con la información de ésta.

Tabla 32. Especificación historia de usuario - Recibo de compra.

Historia de usuario	Listado de donaciones (adm)
Tarjeta	**Como** administrador **quiero** poder ver una lista de todas las donaciones realizadas en la tienda **para** ver información básicas de éstas como la fecha de realización, estado actual, etc., además de poder acceder a una vista específica para cada donación.
Criterios de aceptación	- El usuario debe ser capaz de ver un lista de donaciones - El usuario debe ser capaz de seleccionar una donación de la lista y ser redirigido a una vista donde se encuentra toda la información de esa donación almacenada en el sistema. - El sistema debe devolver todas las donaciones almacenadas en la base de datos.
Escenarios	**Escenario 1**: Visualizar donaciones Dado un administrador del sistema en el panel de administración, cuando entra en la vista de donaciones, entonces debe poder visualizar una lista de donaciones con información muy básica como la fecha de creación, nombre del donante, etc. **Escenario 2**: Acceder a una donación Dado un administrador del sistema en el panel de administración, cuando selecciona una donación en la vista de donaciones, la aplicación le redirige a una vista dónde se encuentra toda la información de la donación seleccionada con todos los datos del sistema.

Tabla 33. Especificación historia de usuario - Listado de donaciones (adm).

Historia de usuario	Información de una donación (adm)
Tarjeta	**Como** administrador **quiero** poder ver la información de una donación en una vista específica **para** ver toda la información que el sistema posee.
Criterios de aceptación	- El usuario debe ser capaz de ver información de la donación. - El sistema ha de devolver información correcta y actualizada.
Escenarios	**Escenario 1**: Visualización de la donación Dado un administrador del sistema en el panel de administración, cuando accede a la vista de la visualización de una donación, la aplicación le muestra al usuario toda la información almacenada sobre la donación.

Tabla 34. Especificación historia de usuario - Información de una donación (adm).

Historia de usuario	Editar donación (adm)
Tarjeta	**Como** administrador **quiero** poder editar la información de una donación en una vista específica **para** poder modificar cambios incorrectos que se hayan podido producir en el proceso de donación por parte del donante.
Criterios de aceptación	- El usuario debe poder realizar cambios a la información de todas las donaciones realizadas. - El usuario ha de ver un comunicado sobre el estado de su edición al acabar. - El sistema debe verificar que los nuevos datos sean correctos. - El sistema ha de ser capaz de actualizar los datos de la donación cambiada.
Escenarios	**Escenario 1**: Edición de una donación con datos válidos Dado un administrador del sistema en el panel de administración en el apartado de donaciones, cuando decide editar una donación con datos correctos, el sistema le muestra un mensaje de confirmación de la edición de la donación. **Escenario 2**: Edición de una donación con datos inválidos Dado un administrador del sistema en el panel de administración en el apartado de donaciones, cuando decide editar una donación con

	datos incorrectos, el sistema le muestra un mensaje de error.

Tabla 35. Especificación historia de usuario - Editar donación (adm).

7.1.3. Regalos

Historia de usuario	Acceso al regalo
Tarjeta	**Como** receptor **quiero** poder acceder a la web a la que hace referencia la carta que he recibido **para** que pueda ver mi regalo a través de un método de acceso no definido aún. Éste método puede ser un código QR o un link que te lleve directamente al regalo o una contraseña para introducir en algún apartado de la aplicación web.
Criterios de aceptación	- El usuario debe ser capaz de acceder a su regalo a través de algún método de acceso. - El sistema ha de verificar de alguna forma el acceso al regalo. - La aplicación ha de mostrar un mensaje cuando se intenta acceder a un regalo inexistente o sin acceso permitido.
Escenarios	**Escenario 1**: Acceso correcto a un regalo existente Dado un receptor de regalo, cuando accede a través de un método de acceso a la vista de su regalo de forma correcta, se le permite obtener la experiencia digital que ofrece la aplicación web (consumo de vídeo, cuestionario, etc.). **Escenario 2**: Acceso incorrecto a un regalo Dado un usuario cualquiera de la aplicación, cuando accede a través de un método de acceso a la vista de su regalo de forma incorrecta, se le mostrará un mensaje de error.

Tabla 36. Especificación historia de usuario - Acceso al regalo.

Historia de usuario	Consumir un vídeo mensaje
Tarjeta	**Como** receptor **quiero** poder consumir un mensaje en forma de video **para** ver lo que hay en él. Éste vídeo será el que un donante se haya encargado de subir para que yo lo reproduzca.
Criterios de aceptación	- El usuario debe ser capaz de reproducir un video mensaje.
Escenarios	**Escenario 1**: Reproducción de vídeo mensaje Dado un receptor de regalo, cuando accede a su regalo de forma

	correcta, se le permite reproducir el vídeo mensaje que el donante ha dejado para él.

Tabla 37. Especificación historia de usuario - Consumir un vídeo mensaje.

Historia de usuario	Notificación de regalo abierto
Tarjeta	**Como** donante **quiero** poder recibir una notificación, en forma de e-mail, cuando el regalo enviado al receptor de la donación ha sido abierto. Esto me servirá **para** saber cuándo mi regalo ha sido recibido.
Criterios de aceptación	- El usuario debe ser capaz de ser notificado vía e-mail en cuanto el receptor de su regalo ha sido abierto. - El sistema debe generar un e-mail en cuanto el receptor accede al regalo.
Escenarios	**Escenario 1**: Notificación vía e-mail Dado un receptor de regalo, cuando accede a su regalo de forma correcta, se generará un e-mail con el donante de la donación como destinatario indicando que el regalo fué abierto.

Tabla 38. Especificación historia de usuario - Notificación de regalo abierto.

Historia de usuario	Marcar regalo como enviado (adm)
Tarjeta	**Como** administrador **quiero** poder marcar un regalo como enviado **para** que los donantes sepan que su regalo ha sido enviada al receptor. Ésto lo podré hacer en el panel de administrador de una forma intuitiva.
Criterios de aceptación	- El usuario debe ser capaz de marcar un regalo como enviado. - Solo se podrán marcar como enviados regalos que no están enviados.
Escenarios	**Escenario 1**: Marcar regalo como enviado Dado un administrador del sistema en el panel de administración, cuando accede a la vista de la visualización de una donación, el administrador puede marcar el regalo asociado como enviado al receptor.

Tabla 39. Especificación historia de usuario - Marcar regalo como enviado (adm).

Historia de usuario	Descargar carta (adm)

Tarjeta	**Como** administrador **quiero** poder descargar la carta que recibirán los receptores en formato PDF **para** que pueda enviarlo por correo.
Criterios de aceptación	- El usuario de descargar un pdf a través de un enlace / botón. - El sistema debe poder generar el pdf con información correcta del regalo.
Escenarios	**Escenario 1**: Descargar carta Dado un administrador del sistema en el panel de administración, cuando accede a la vista de la visualización de una donación, el administrador puede descargar una carta para el receptor que contiene el acceso al regalo.

Tabla 40. Especificación historia de usuario - Descargar carta (adm).

7.1.4. Cuestionarios

Historia de usuario	Listado de cuestionarios (adm)
Tarjeta	**Como** administrador **quiero** poder ver una lista de todos los cuestionarios introducidos en el sistema con información básica como número de preguntas, título, etc. **para** tener una visión general sobre éstos.
Criterios de aceptación	- El usuario debe ser capaz de ver un lista de cuestionarios - El usuario debe ser capaz de seleccionar un cuestionario de la lista y ser redirigido a una vista donde pueda editar el cuestionario. - El sistema debe devolver todos los cuestionarios almacenados en la base de datos.
Escenarios	**Escenario 1**: Visualizar cuestionarios Dado un administrador del sistema en el panel de administración, cuando entra en la vista de cuestionarios, entonces debe poder visualizar una lista de cuestionarios con información superficial como el número de preguntas. **Escenario 2**: Acceder a un cuestionario Dado un administrador del sistema en el panel de administración, cuando selecciona un cuestionario en la vista de cuestionarios, la aplicación le redirige a una vista dónde se encuentra toda la información del cuestionario.

Tabla 41. Especificación historia de usuario - Listado de cuestionarios (adm).

Historia de usuario	Crear cuestionario (adm)
Tarjeta	**Como** administrador **quiero** poder crear cuestionarios **para** introducirlos en la aplicación y pueda ser enlazado a cualquier proyecto de caridad.
Criterios de aceptación	- El usuario debe ser capaz de introducir preguntas y respuestas a través de un formulario. - El sistema debe verificar que los datos sean correctos. - El sistema ha de ser capaz de registrar el cuestionario. - El usuario ha de ver un comunicado sobre el estado de la creación del cuestionario.
Escenarios	**Escenario 1**: Creación de un cuestionario con datos válidos Dado un administrador del sistema en el panel de administración, cuando decide crear un nuevo cuestionario y rellena una serie de formularios que representan las preguntas y respuestas, la aplicación le confirma la creación del cuestionario y le redirige a una vista con los datos de la creación del cuestionario. **Escenario 2**: Creación de un cuestionario con datos inválidos Dado un administrador del sistema en el panel de administración, cuando decide crear un nuevo cuestionario y rellena una serie de formularios que representan las preguntas y respuestas de forma errónea (como por ejemplo marcar dos respuestas para una pregunta como correctas), la aplicación muestra un mensaje de error y le permite corregir los fallos.

Tabla 42. Especificación historia de usuario - Crear cuestionario (adm).

Historia de usuario	Eliminar cuestionario (adm)
Tarjeta	**Como** administrador **quiero** poder eliminar cuestionarios **para** que no se pueda enlazar a proyectos de caridad.
Criterios de aceptación	- El usuario debe ser capaz de eliminar un cuestionario que no esté vinculado a ningún proyecto de caridad. - El sistema debe verificar que el cuestionario se pueda eliminar. - El usuario ha de ver un comunicado sobre el estado de la eliminación del cuestionario.

Escenarios	**Escenario 1**: Eliminación de un cuestionario no vinculado a ningún proyecto de caridad Dado un administrador del sistema en el panel de administración, cuando decide eliminar un cuestionario que no está vinculado a ningún proyecto de caridad, la aplicación le redirige a la lista de cuestionarios y le muestra un mensaje que confirma la eliminación del cuestionario. **Escenario 2**: Eliminación de un cuestionario vinculado a al menos un proyecto de caridad Dado un administrador del sistema en el panel de administración, cuando decide eliminar un cuestionario que está vinculado a al menos un proyecto de caridad, la aplicación le muestra un error.

Tabla 43. Especificación historia de usuario - Eliminar cuestionario (adm).

Historia de usuario	Editar cuestionario (adm)
Tarjeta	**Como** administrador **quiero** poder editar un cuestionario **para** actualizar las preguntas y respuestas que tiene.
Criterios de aceptación	- El usuario debe ser de editar las preguntas y respuestas del cuestionario. - El sistema debe verificar que los cambios sean correctos. - El sistema ha de ser capaz de editar el cuestionario. - El usuario ha de ver un comunicado sobre el estado de su edición al acabar.
Escenarios	**Escenario 1**: Edición de un cuestionario con cambios válidos Dado un administrador del sistema en el panel de administración, cuando decide editar un cuestionario com datos correctos en las preguntas, respuestas o información del cuestionario, el sistema le muestra un mensaje de confirmación de la edición del cuestionario. **Escenario 2**: Edición de un cuestionario con cambios inválidos Dado un administrador del sistema en el panel de administración, cuando decide editar un cuestionario con datos incorrectos en las preguntas, respuestas o información del cuestionario, el sistema le muestra un mensaje de error.

Tabla 44. Especificación historia de usuario - Editar cuestionario (adm).

Historia de usuario	Consumir cuestionario
Tarjeta	**Como** receptor **quiero** poder consumir un cuestionario con diversas preguntas **para** concienciarme sobre el tema del cuestionario.
Criterios de aceptación	- El usuario debe ser capaz de consumir un cuestionario. - La aplicación web debe mostrar los aciertos y errores del cuestionario.
Escenarios	**Escenario 1**: Realización de cuestionario Dado un receptor de un regalo, cuando accede a su regalo de forma correcta, se le permite realizar un cuestionario que contiene al menos un pregunta y diversas respuestas. Una vez conteste una pregunta, se le mostrará si ha sido un error o acierto. Al final del cuestionario podrá cambiar de vista para consumir otra parte de su regalo como por ejemplo el video mensaje.

Tabla 45. Especificación historia de usuario - Consumir cuestionario.

7.1.5. Otras tareas

Historia de usuario	Información estática sobre la web
Tarjeta	**Como** visitante **quiero** leer información general sobre la web **para** saber más de ésta, tales como la finalidad, cómo funciona, contacto, etc.
Criterios de aceptación	- El usuario debe ser capaz de ver un apartado que explique cómo funciona la web. - El usuario debe ser capaz de ver un apartado con los términos del servicio. - El usuario debe ser capaz de ver un apartado que habla sobre la privacidad del servicio (datos de usuarios, regalos, etc.).
Escenarios	**Escenario 1**: Acceso a la vista principal de la aplicación Dado un usuario de la aplicación, cuando accede a la vista principal de la aplicación web, se encuentra con diversos apartados como la finalidad del servicio ofrecido, cómo funciona. También proporciona acceso a apartados donde se hallan los términos de la aplicación, así como la privacidad.

Tabla 46. Especificación historia de usuario - Información estática sobre la web.

Historia de usuario	Invitación al panel de administración
Tarjeta	**Como** administrador **quiero** poder añadir a personas al panel de administración a través de una invitación **para** que pueda empezar a administrar la aplicación.
Criterios de aceptación	- El usuario debe ser capaz de invitar a alguien al panel de administración a partir de un e-mail. - El usuario receptor de la invitación al panel de administrador debe poder introducir una contraseña a través de una forma en la invitación. - El sistema debe registrar al nuevo administrador en el momento de la invitación, pero sin acceso hasta que acepte la invitación. - El sistema debe actualizar actualizar la contraseña del nuevo administrador una vez haya aceptado la invitación y haya introducido una contraseña.
Escenarios	**Escenario 1**: Invitar a usuario que acepta la entrada Dado un administrador en el panel de administración, cuando invita a un nuevo administrador introduciendo su e-mail y éste acepta la invitación introduciendo una contraseña. Entonces el nuevo administrador debe poder empezar a usar el panel de administración con su nueva forma de acceso. **Escenario 2**: Invitar a usuario que no acepta la entrada Dado un administrador en el panel de administración, cuando invita a un nuevo administrador introduciendo su e-mail y no éste acepta la invitación introduciendo una contraseña. Entonces el nuevo administrador no debe tener forma de acceder al panel de administración.e

Tabla 47. Especificación historia de usuario - Invitación al panel de administración.

7.2. Esquema conceptual de los datos

El esquema conceptual de los datos es un diagrama en UML[24] que representan los conceptos que caracterizan al sistema [15]. Los elementos generales que contiene son:

- **Clases de objetos:** Son las entidades que describen un conjunto de objetos. Estos objetos tienen una semántica común, las mismas propiedades, mismas relaciones con otros objetos. Básicamente un comportamiento común.

[24] Son las siglas de 'Lenguaje Unificado de Modelado'. "*Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.*" [14]

- **Atributos:** Son características que tienen las clases de objetos antes mencionadas. Estas propiedades son compartidas por todos los objetos de una clase.

- **Asociaciones:** Indican las relaciones entre dos o más objetos.

- **Restricciones:** Son las reglas que no pueden ser representadas gráficamente en UML. Es por ello que se suelen especificar de forma textual.

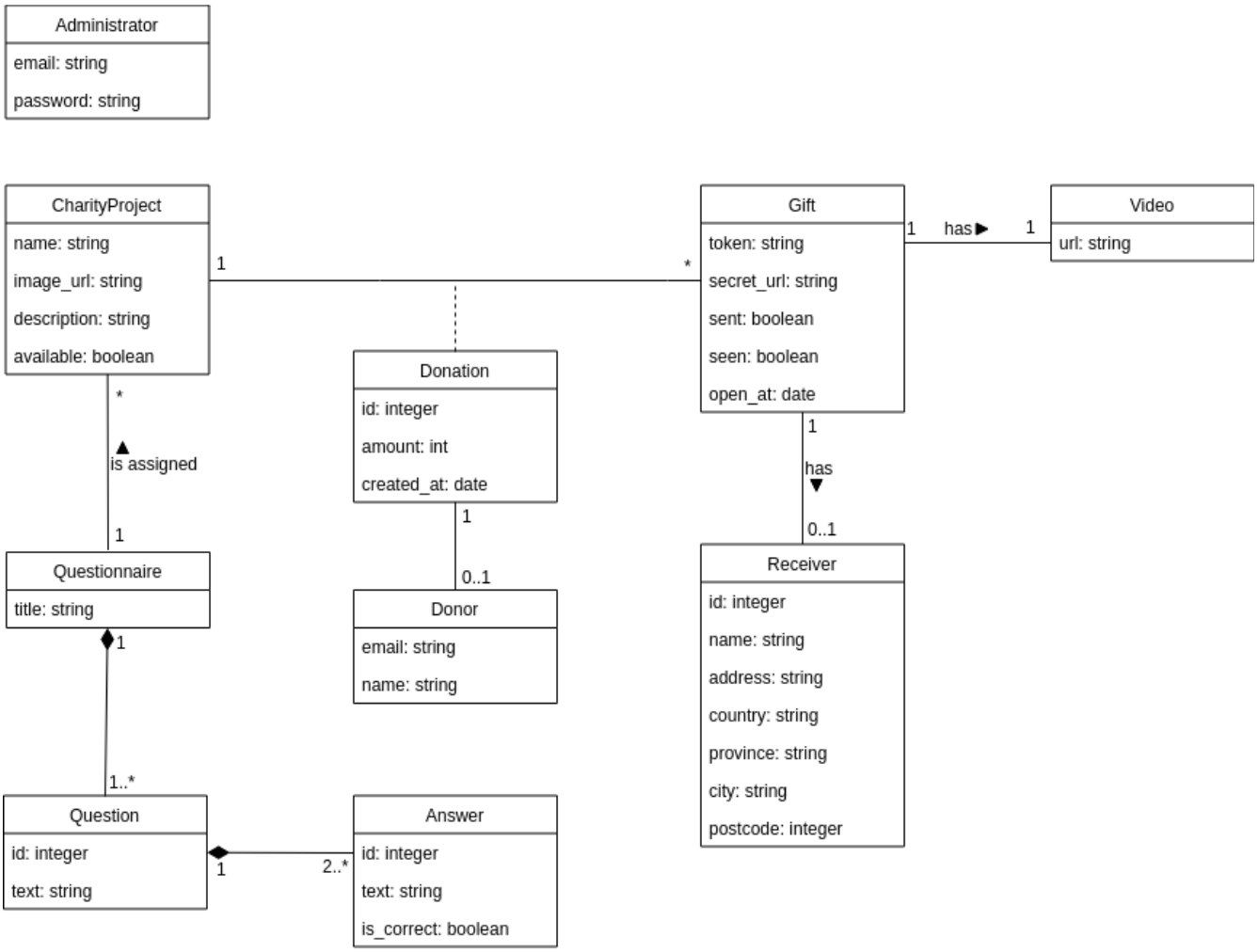


Imagen 2. Esquema conceptual.

Restricciones textuales

1. Claves externas: (Administrator, email), (CharityProject, name), (Questionnaire, title), (Question, id), (Answer, id), (Donor, id), (Gift, token), Receiver(id), Video(url), (Donation, id).
2. Un regalo no puede estar marcado como 'visto' si no ha sido enviado a su receptor.
3. La fecha de visualización de un regalo ha de ser posterior a la fecha de la creación de la donación a la que pertenece.
4. No pueden haber dos respuestas marcadas como correctas que pertenezcan a la misma pregunta.
5. No pueden haber dos respuestas iguales que pertenezcan a la misma pregunta.
6. No pueden haber dos preguntas iguales que pertenezcan al mismo cuestionario.

7.2.1. Descripción de las clases

Administrador (Administrator): Esta clase representa a las personas que controlarán la aplicación desde el panel de administración. Tiene un e-mail (email) que lo identifica y una contraseña (password) que serán sus datos de acceso.

Proyecto de caridad (Charity project): Esta clase representa a los proyectos sin ánimo de lucro al que los donantes podrán donar. Tiene un nombre (name) que lo identifica, una dirección web a una imagen que refleja el proyecto (image_url), una descripción y una marca que indica si está o no disponible en la aplicación web (available).

Cuestionario (Questionnaire): Esta clase representa los cuestionarios que están asignados a los proyectos de caridad. Tiene un título que lo identifica y como mínimo una pregunta asociada.

Pregunta (Question): Esta clase representa las preguntas que forman parte de un cuestionario. Tiene una id que lo identifica, un texto (text) que plantea la pregunta y varias posibles respuestas.

Respuesta (Answer): Esta clase representa las respuestas que forman parte de un cuestionario. Tiene una id que lo identifica, un texto (text) que es la respuesta en sí y una marca que indica si es correcta o no (is_correct).

Donante (Donor): Esta clase representa a los donantes que realizan donaciones. Tiene una id que lo identifica, un e-mail (email) y un nombre (name).

Regalo (Gift): Esta clase representa los regalos que recibirán los donantes. Tiene un token que lo identifica, una marca que indica si ha sido enviado o no (sent), otra marca que representa si ha sido visto por el receptor del regalo o no (seen), una dirección web secreta (secret_url) para que el receptor pueda visualizar el regalo, una fecha que indica cuándo ha sido abierto por el receptor (open_at). También tiene un vídeo relacionado y un receptor, aunque la información de éste último puede ser borrado debido al requisito no funcional de cumplimiento legal que tiene el sistema, es por ello que los regalos se pueden quedar sin receptor.

Receptor (Receiver): Esta clase representa los receptores que reciben regalos. Tiene una id que lo identifica, un nombre (name), una dirección (address), país (country), provincia (province), ciudad (city) y un código postal (postcode)

Vídeo (Video): Esta clase representa los vídeos que hay en el sistema y que pertenecen a regalos. Tiene una id que lo identifica y una dirección web donde se encuentra el vídeo en cuestión (url).

Donación (Donation): Esta clase representan las donaciones del sistema. Tiene una id que lo identifica, una cantidad de dinero donada (amount), y una fecha de realización de la donación (created_at). Esta clase será el resultado del proceso de donación que tendrá el sistema y creará un objeto de la clase donante (que también puede ser borrado posteriormente al igual que los receptores), y otro de la clase

regalo. Cada vez que un usuario de la aplicación realice una donación, además de crear los objetos antes dichos, se le asociará el proyecto de caridad del cual se está haciendo la donación,

8. Diseño

Después de haber realizado el análisis de requisitos y la especificación, es hora de hablar del diseño que ha tenido el sistema. Para ello, se ha decidido aplicar el patrón 'Arquitectura en tres capas' de manera que el sistema pueda desacoplarse y los cambios en el desarrollo sólo afecten al nivel en el cual se haya hecho [16].

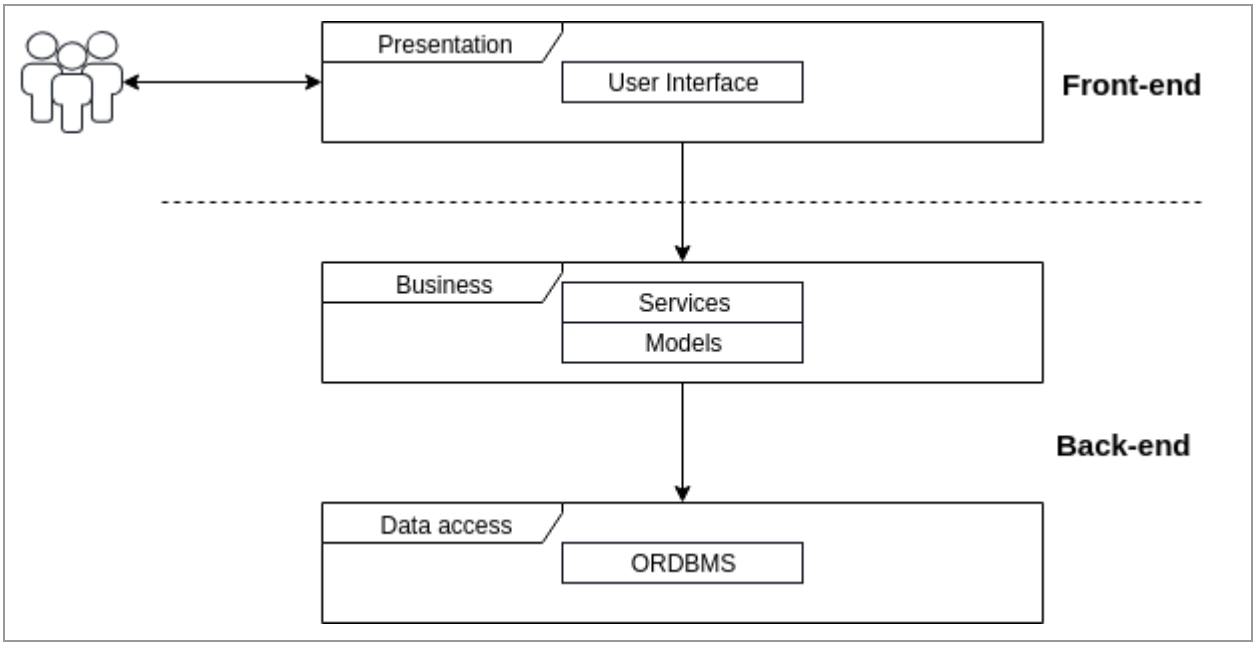


Imagen 3. Arquitectura en tres capas.

La idea de haber realizado esta aplicación con este tipo de arquitectura de tres capas es para que además de minimizar los efectos colaterales debido a posibles errores que se introduzcan, se pueda hacer una aplicación extendible. Ahora mismo la idea fue desarrollar una aplicación web, pero si en un futuro se plantease desarrollar una aplicación móvil nativa, se podría hacer perfectamente gracias a la manera en la que se construirá la aplicación. Además, se destinarían menos recursos, pues solo se tendría que cambiar la capa que se dedique a la interfaz gráfica.

Como comentario final, decir que para realizar esta aplicación se han hecho dos proyectos por separado. Uno que se ha encargado solamente de la capa de presentación (front-end), y otro de la capa de negocios y de datos (back-end).

8.1. Infraestructura

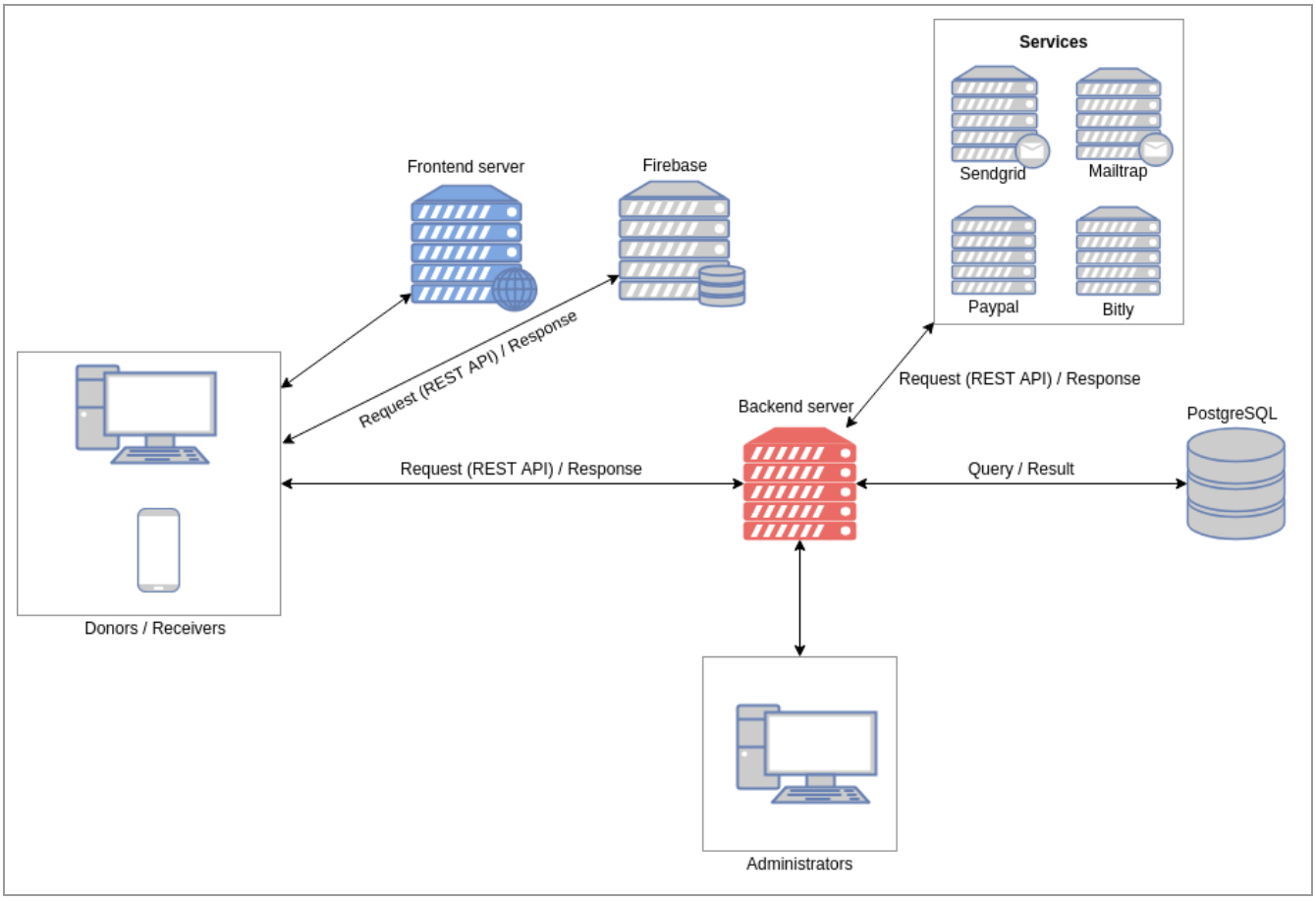


Imagen 4. Infraestructura de la aplicación.

Antes de profundizar en las diferentes capas que tiene la aplicación realizada, hablaremos sobre la infraestructura que ha tenido la aplicación. Tal y como vemos en la imagen 4, la infraestructura está constituida por:

Servidor front-end: Este servidor almacenará el front-end de la aplicación y tal y como comenté en el capítulo de metodología, he usado Heroku para almacenar éste proyecto. Este servidor almacena un proyecto que usa un framework para desarrollar las vistas de la aplicación. El usuario, con un cliente como un móvil o un ordenador, podrá comunicarse con éste servidor a través de un navegador web, para así obtener la información de las vistas de la aplicación y ser interpretado por el mismo navegador web.

También podemos ver una conexión con el servicio Firebase, que se encarga de almacenar los vídeos descritos en la historia de usuario *Subir vídeo*.

Servidor back-end: Para este servidor, también he usado Heroku y contiene un proyecto con un framework MVC[25]. Éste servidor recibe peticiones HTTP[26] para responder de forma correcta a las peticiones, ya que tiene toda la lógica del sistema. Además puede realizar otras llamadas a las API[27] de servicios de terceros o comunicarse con el ORDBMS, que estará situado en otro servidor.

Cabe mencionar que en el panel de administrador se situará también en el back-end, debido a que para su construcción, se ha usado una librería que acelere su construcción. Hoy en día, muchos de los frameworks back-end tiene alguna forma de producir un panel de administrador de manera rápida y personalizable. La idea es hacer uso de algún recurso de este estilo con tal de no tener que escribir las operaciones CRUD[28], sino dedicarme a personalizarlo y a añadir operaciones adicionales como la descarga de la carta para los receptores de regalos.

Servicios: En la imagen 4 he representado a los servicios como un cúmulo de servidores distintos que se comunican a través de una API con el servidor back-end. Cada servicio utilizado se comunica con un servidor diferente pues normalmente cada servicio es ofrecido por empresas distintas especializadas en lo suyo.

Sistema de administración de base de datos: Esta base de datos objeto - relacional estará situada en otro servidor separado. Haciendo uso de un complemento de Heroku llamado Heroku Postgres [20], puedo obtener el acceso a una base de datos de forma rápida, sin necesidad de configurarlo y estando situado en un servidor al cual no tengo acceso directo. Este complemento de Heroku me permite obtener credenciales de acceso a la base de datos y así configurar el servidor back-end para que haga uso de él.

8.2. Capa de presentación

Esta capa contendrá la interfaz gráfica. Capturará la información del usuario y se comunicará con la capa de negocio. La idea desde el principio fue que haya un servidor que aloje solo las vistas de la aplicación.

Actualmente existen muchas herramientas que permiten trabajar con sólo la interfaz de usuario, de manera que desarrollando un proyecto con un framework front-end, podemos desacoplar esta capa de la de negocios y éstas se pueden comunicar a través de un protocolo de comunicación HTTP.

---

[25] Es un conjunto de librerías y convenciones que sigue un patrón arquitectónico de software Modelo-Vista-Controlador, en el que la aplicación software se divide en tres partes interconectadas.
[26] Acrónimo de: Hypertext Transfer Protocol (Protocolo de Transferencia de Hiper Textos). Es un protocolo que trata las peticiones de un cliente y las respuestas de un servidor en la Internet. [17]
[27] Acrónimo de: Application Programming Interface (Interfaz de Programación de Aplicaciones). "*Es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas.*" [18]
[28] Acrónimo de: Create, Read, Update y Delete (Crear, Leer, Actualizar y Borrar). *"Se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software."* [19]

8.2.1. Descripción de las pantallas

En esta sección se mostrarán todas las vistas importantes de la aplicación divididas en dos partes. Por un lado, se encuentran las vistas de la aplicación principal, que es lo que los donantes y receptores usan. Mientras que por el otro lado, tenemos el panel de administrador, gestionado por los administradores.

Aplicación principal

- Vista principal:

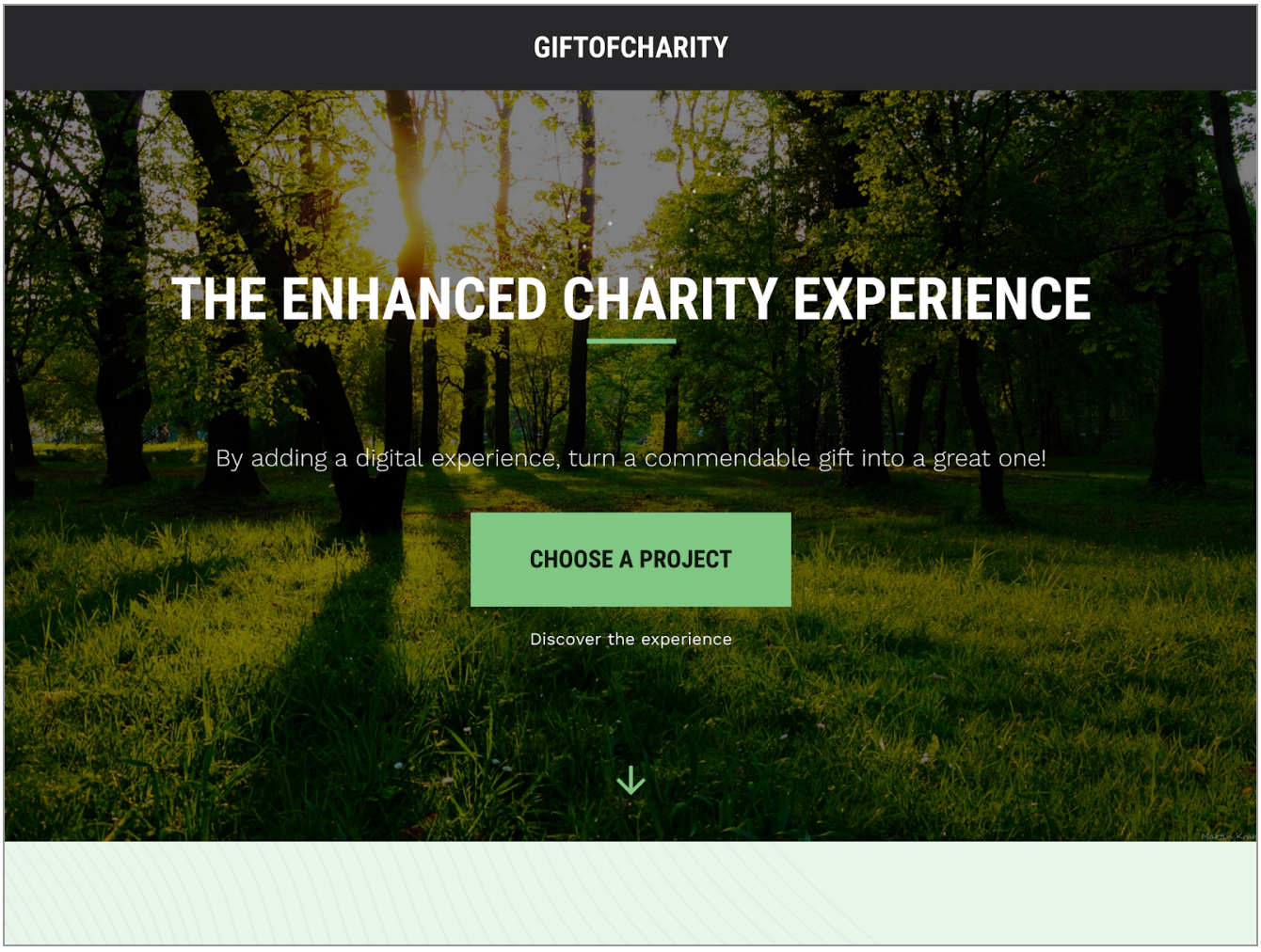


Imagen 5. Vista principal.

En la imágen 5 se ve la vista principal de la aplicación web. Tiene un diseño minimalista donde predominan los verdes y dispone de un botón que hace que el navegador se centre en los proyectos de caridad a elegir.

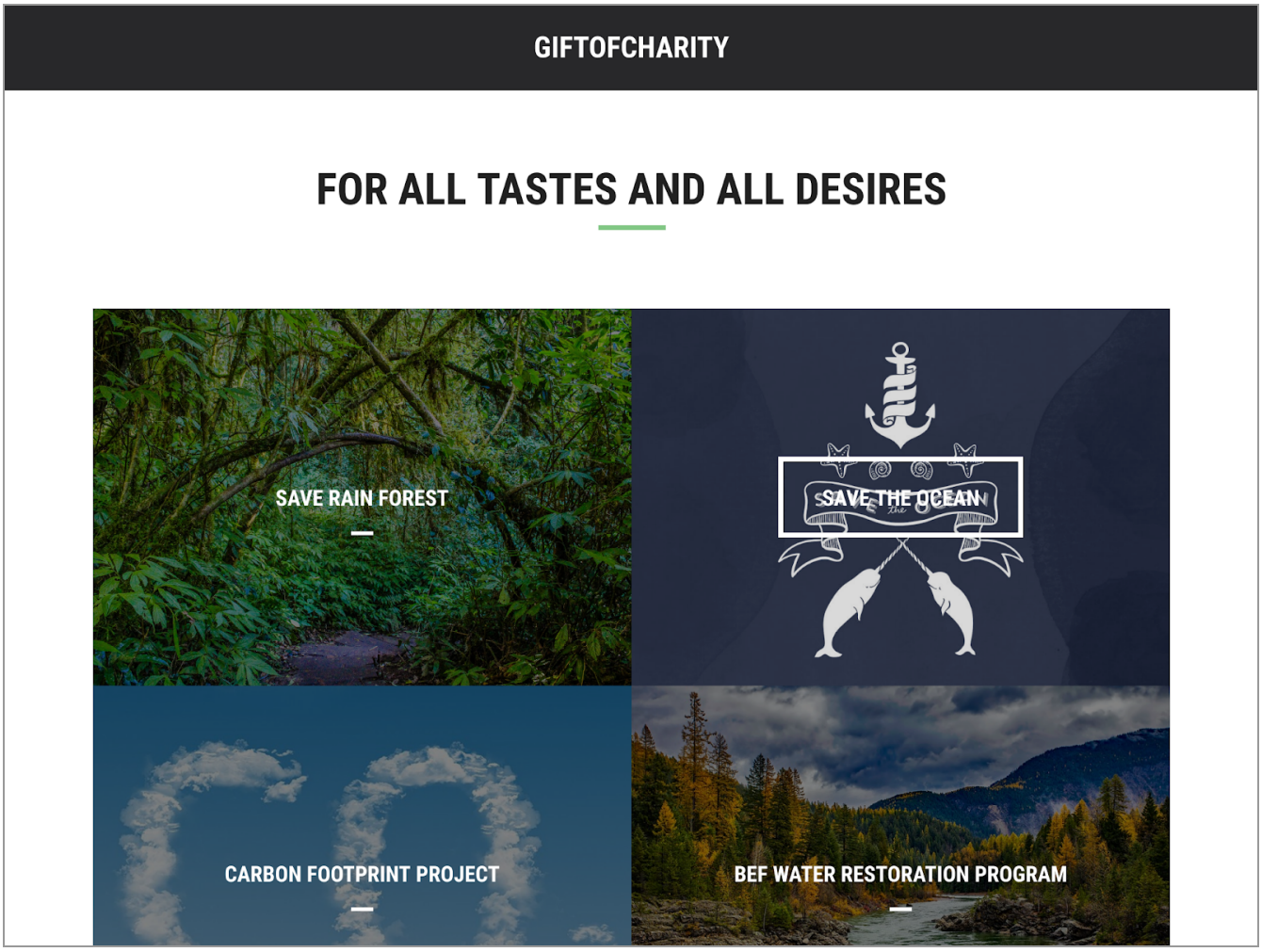


Imagen 6. Proyectos de caridad a elegir.

La imágen 6 es una parte de la vista principal. No se ha creado otra vista en específico para elegir proyectos de caridad, sino que se ha unificado se manera que desde la vista principal se pueda escoger entre diversos proyectos de caridad. Éstos contienen una imagen y el nombre del proyecto en sí, que al ser pulsados la vista cambia a la del producto pulsado.

- Vista de los proyectos de caridad:

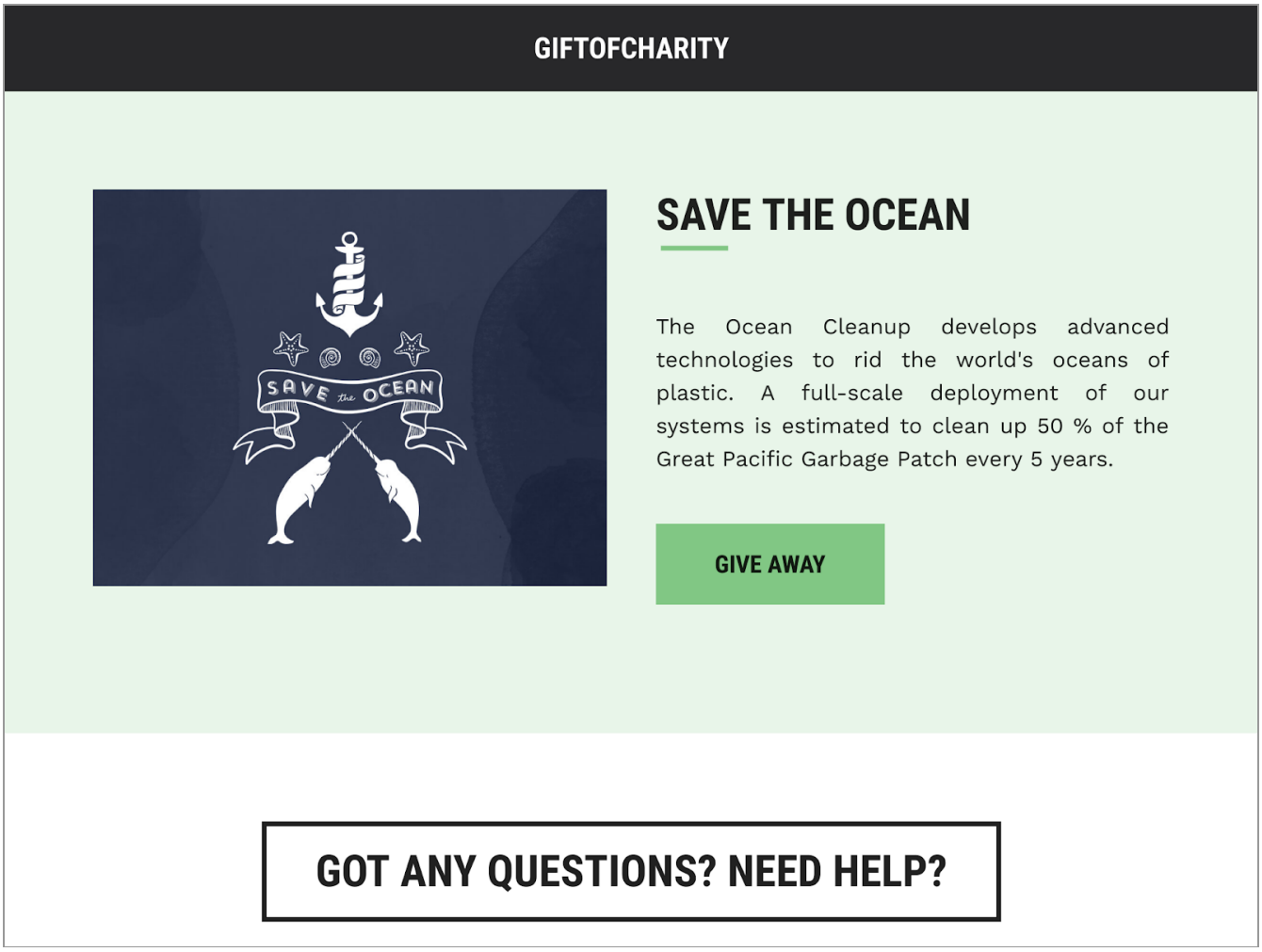


Imagen 7. Vista de un proyecto de caridad.

Cada proyecto de caridad dispone de una vista propia, de tal manera que se puede leer el nombre del proyecto de caridad, una imágen y una pequeña descripción. Pulsando el botón ‘Give away’, la aplicación muestra la vista del proceso de donación.

Abajo podemos ver un botón en grande que suele estar presente en algunas vistas de la aplicación. El botón mostrado es un link hacia el e-mail oficial de la aplicación web.

- Vistas del proceso de donación:

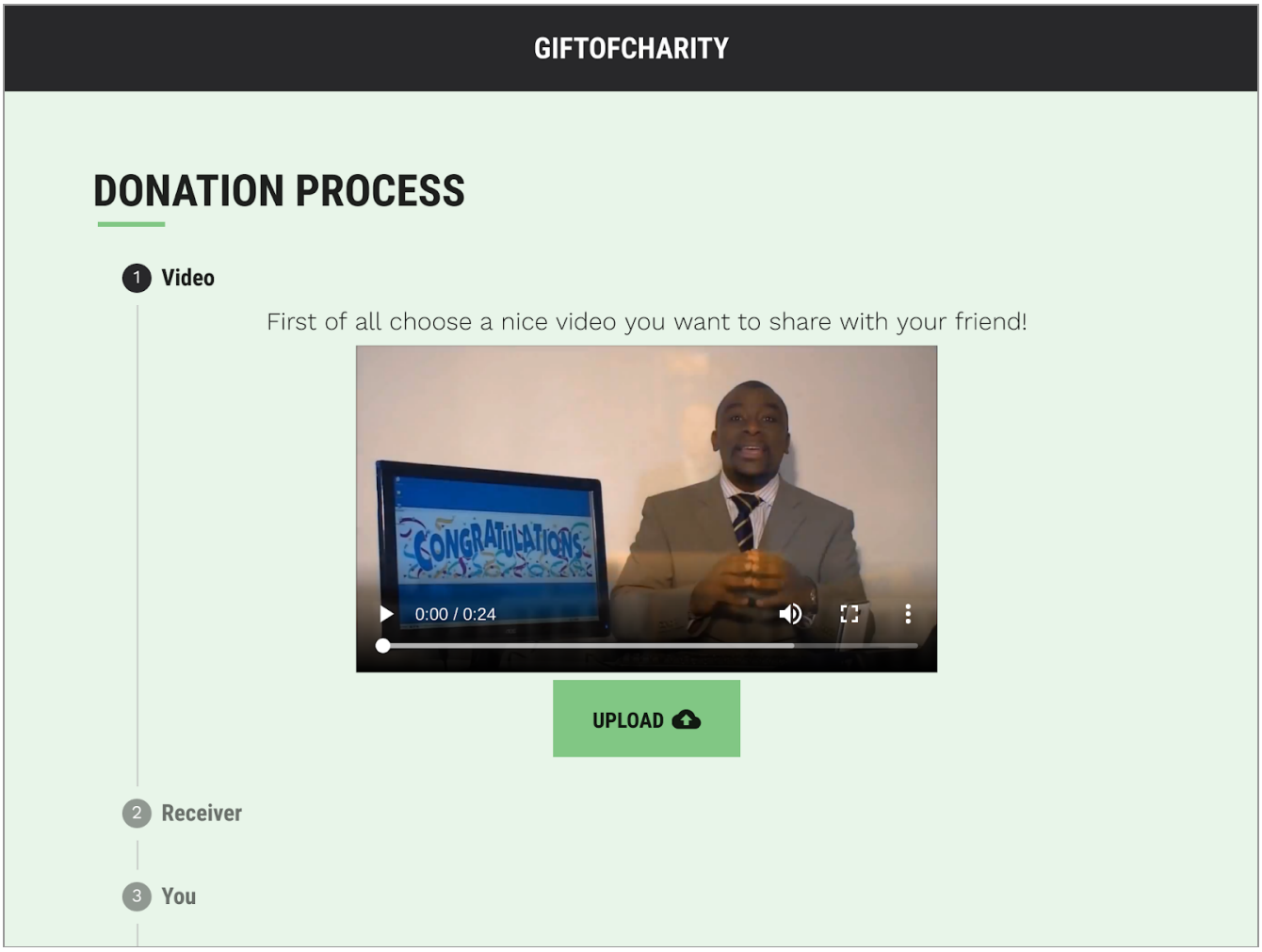


Imagen 8. Vista del proceso de donación - Vídeo.

La vista del proceso de donación dispone de 6 subvistas. En la imágen 8 podemos ver la primera, en la que un donante ya ha subido un vídeo suyo a través del botón 'Upload'. Como podemos observar, los vídeos subidos se pueden previsualizar antes de pasar al siguiente paso.

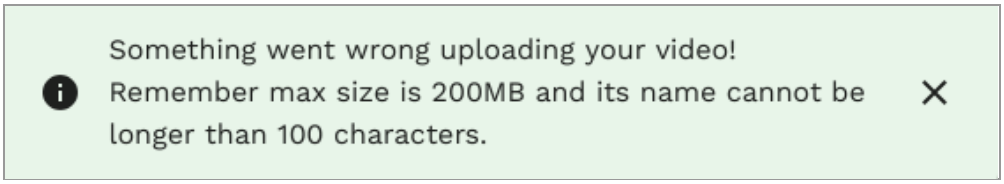


Imagen 9. Vista del proceso de donación - Mensaje de error.

Si no se sube ningún vídeo, el proceso de donación no deja pasar al siguiente paso. Además, muestra un mensaje de error si se intenta subir un archivo que no es un vídeo.

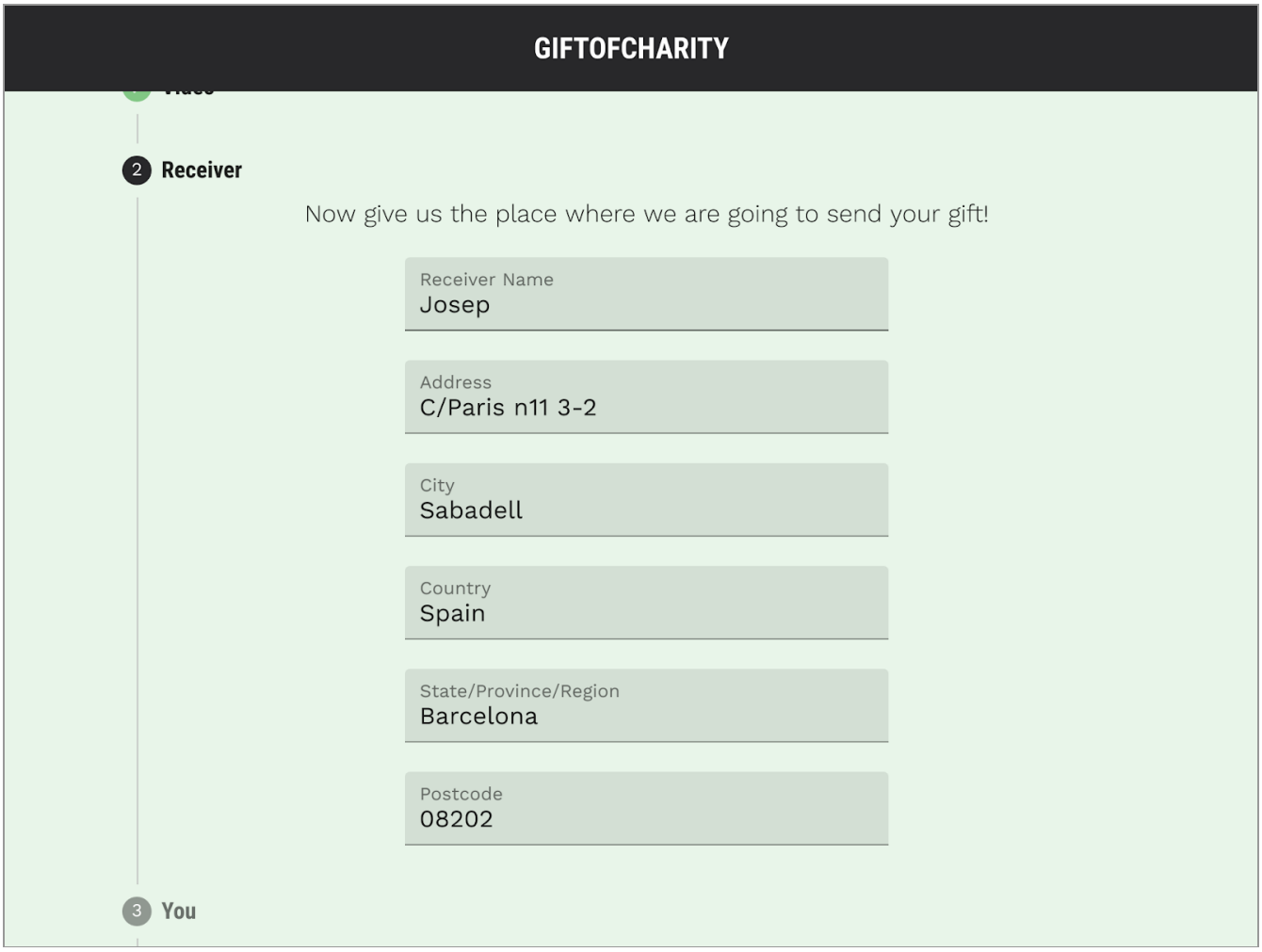


Imagen 10. Vista del proceso de donación - Receptor.

En la imágen 10 se encuentra el segundo paso del proceso de donación. En esta subvista, el usuario donante tiene que introducir los datos del receptor a través de un formulario que cuenta con unos comprobadores de formato, de manera que si hay algún dato erróneo, la entrada se marca con rojo.

Si el formulario cuento con al menos un dato erróneo. No se puede pasar al siguiente paso del proceso de donación.

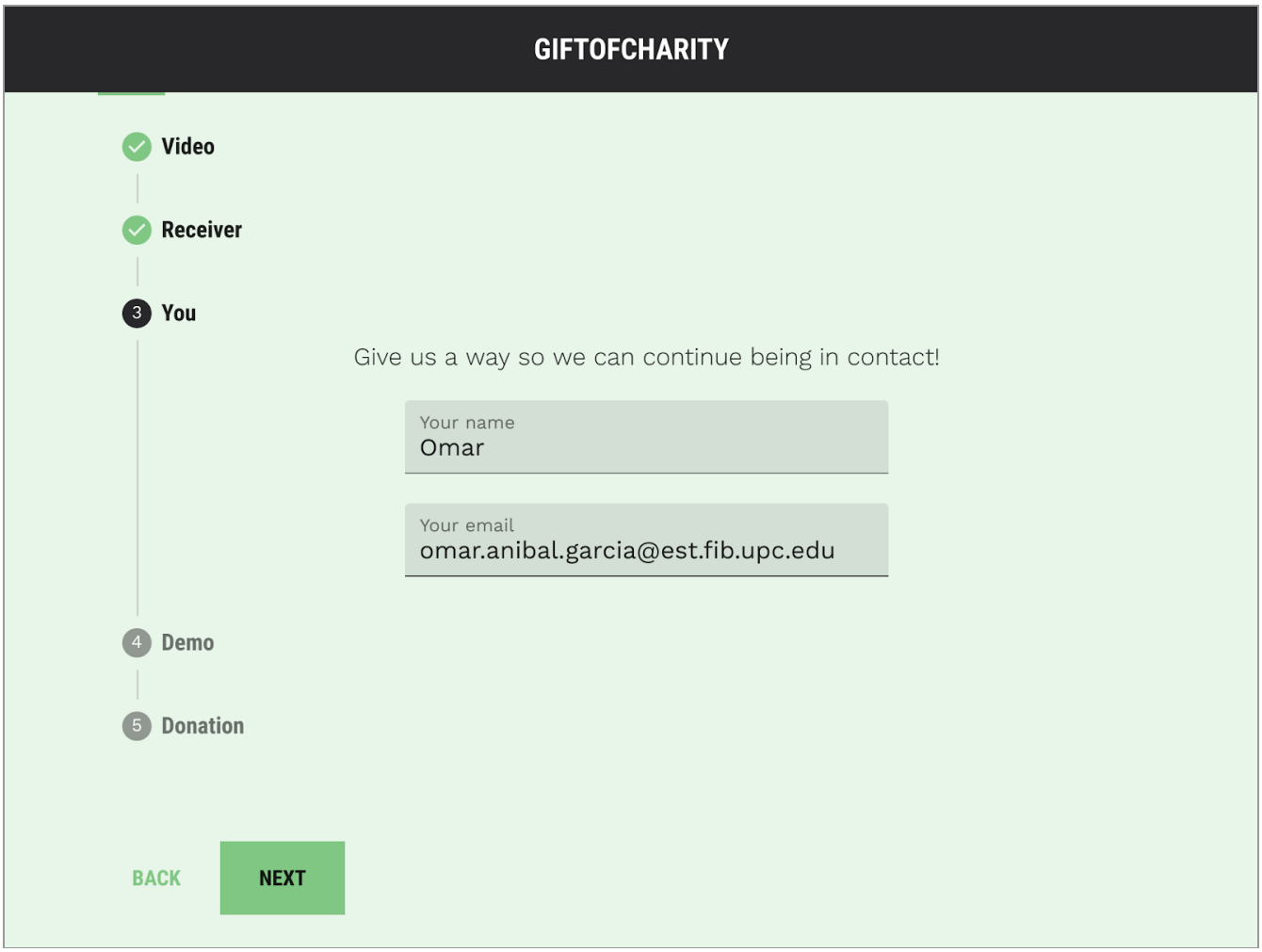


Imagen 11. Vista del proceso de donación - Donante.

En la imagen 11, vemos el paso del proceso de donación donde el usuario donante introduce sus propios datos, que básicamente son un nombre y un e-mail.

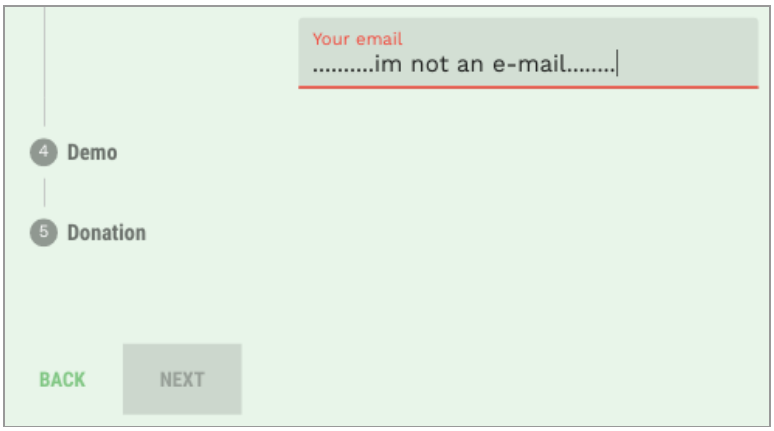


Imagen 12. Proceso de donación - Campo de formulario erróneo.

De manera simétrica al paso anterior, si hay un error, el campo se marca en rojo. Y si hay algún campo en rojo, el botón de 'next' no se muestra en verde para poder ser pulsado.

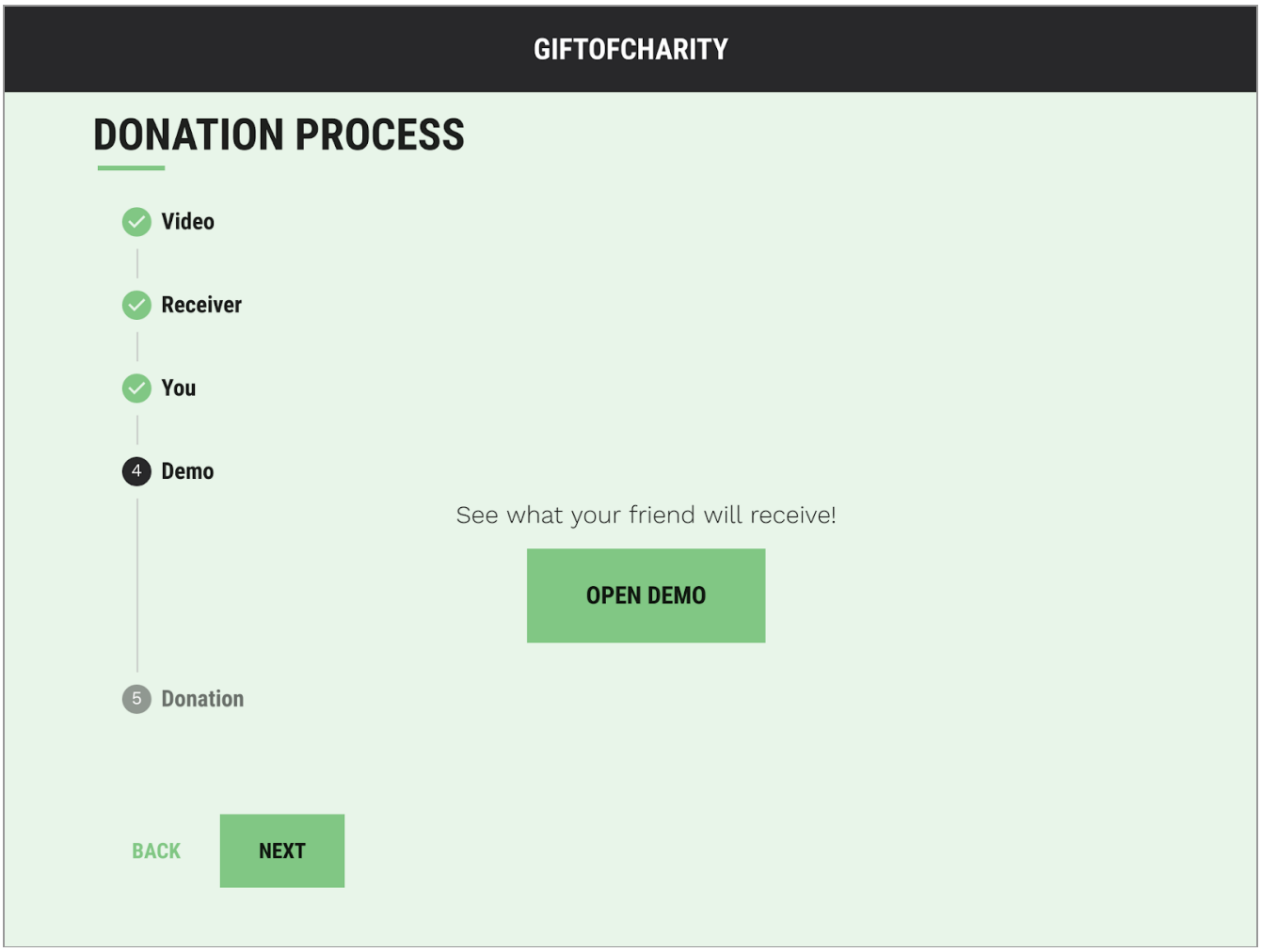


Imagen 13. Vista del proceso de donación - Demo.

En la imagen 13 podemos ver que hay un paso en el que el usuario donante puede pulsar un botón para que se abra la demo del regalo del receptor, que básicamente es una nueva pestaña en el navegador, que contiene el regalo compuesto por el cuestionario y el vídeo mensaje a través de un enlace de un solo uso.

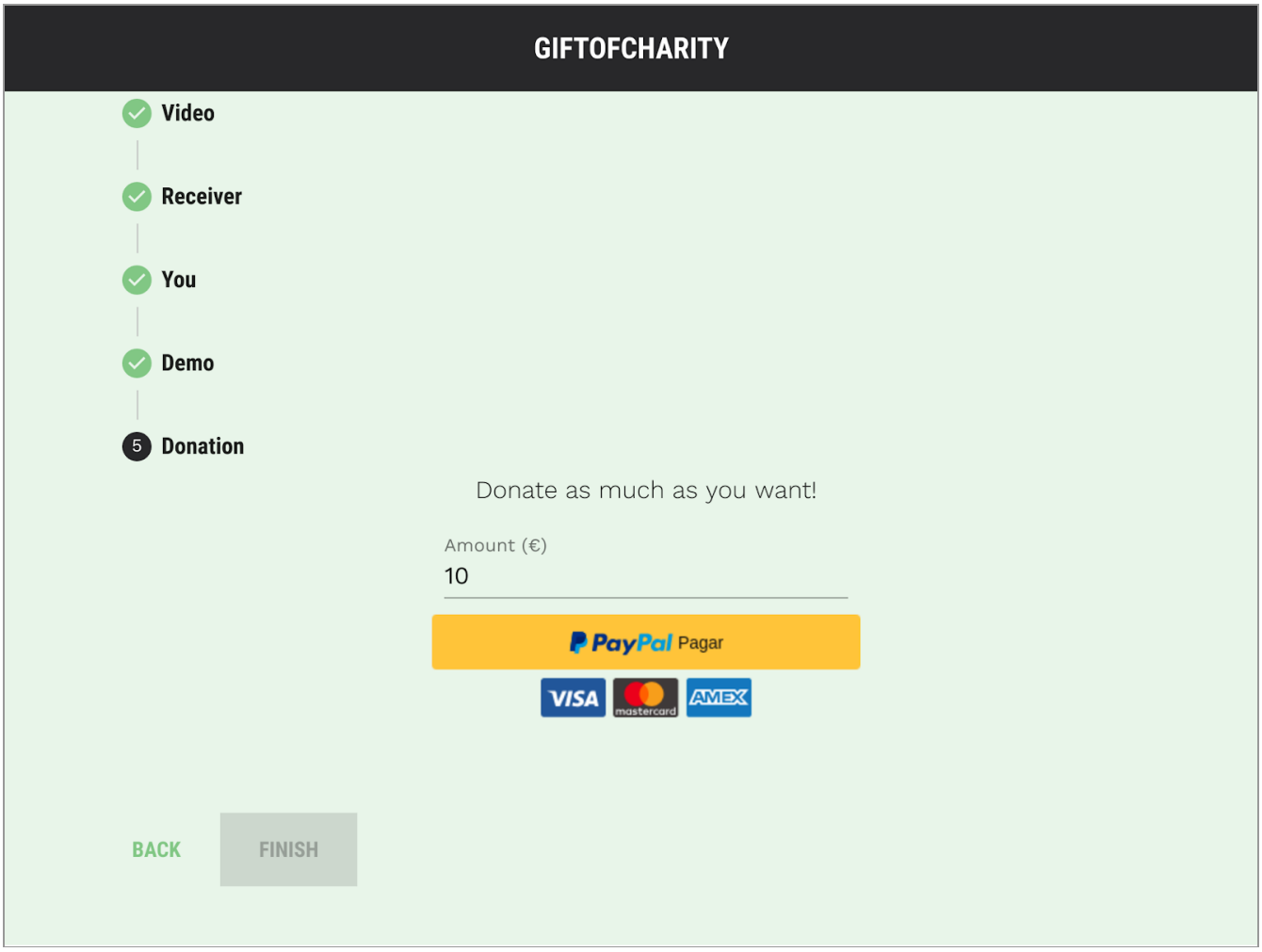


Imagen 14. Vista del proceso de donación - Pago..

En la imagen vemos el último paso del proceso de donación, donde el usuario donante puede introducir la cantidad a donar con un número entero y un mínimo de 1 €. Si se introduce un número erróneo, el campo se marca en rojo y el botón de pagar desaparece.

Cuando el usuario pulsa el botón de paypal, se abre un pop up donde el usuario puede pagar. Si el pago ha ido mal, se muestra un mensaje de error igual que cuando se sube un vídeo erróneo y el usuario puede volver a intentar donar.

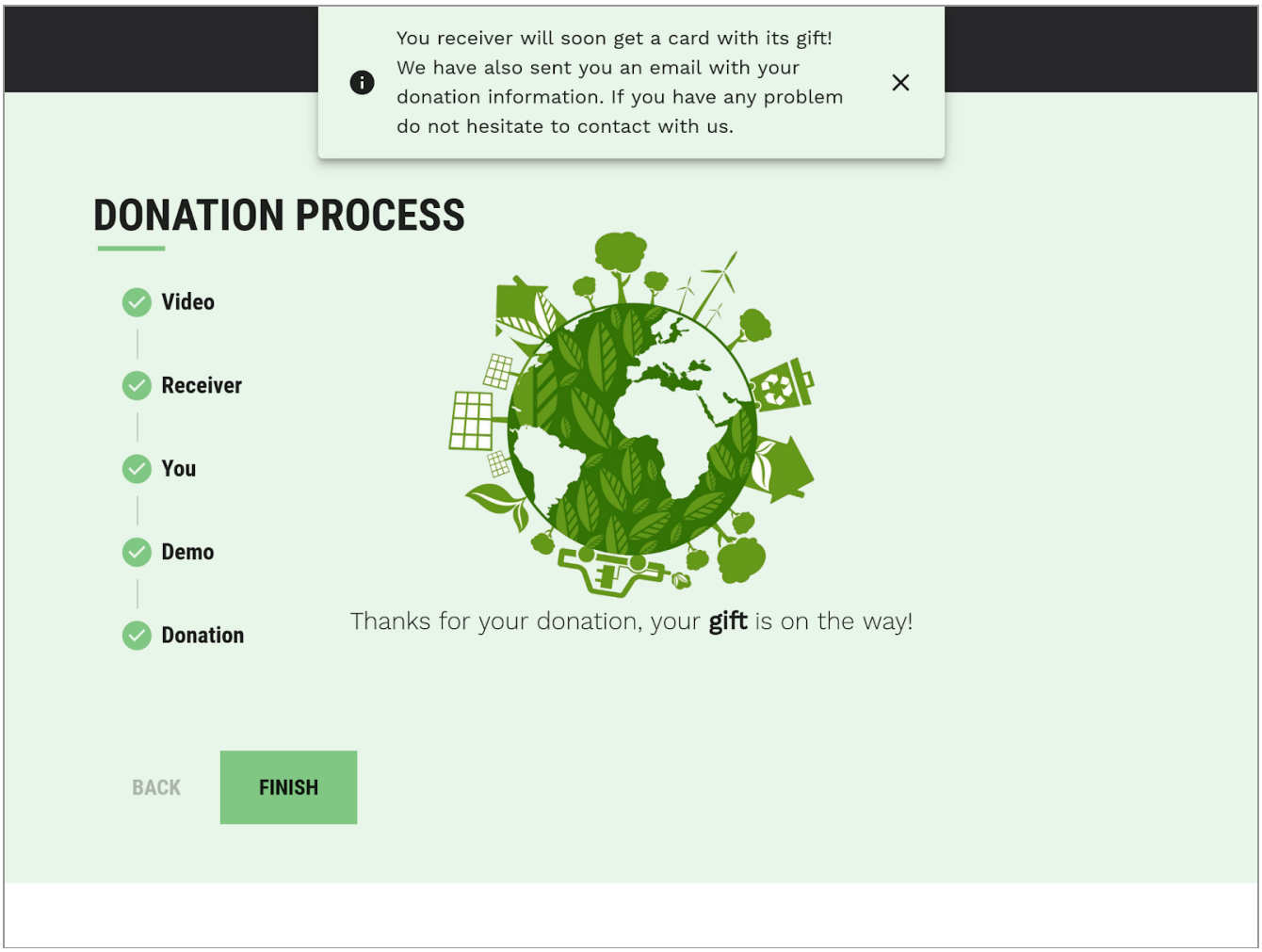


Imagen 15. Vista de proceso de donación - Mensaje final.

Finalmente en la imagen 15 tenemos el mensaje que se muestra una vez la donación ha sido hecho correctamente. Como podemos ver, hay un botón de 'Finish' que lleva de vuelta a la vista principal de la aplicación.

- Vistas del cuestionario:

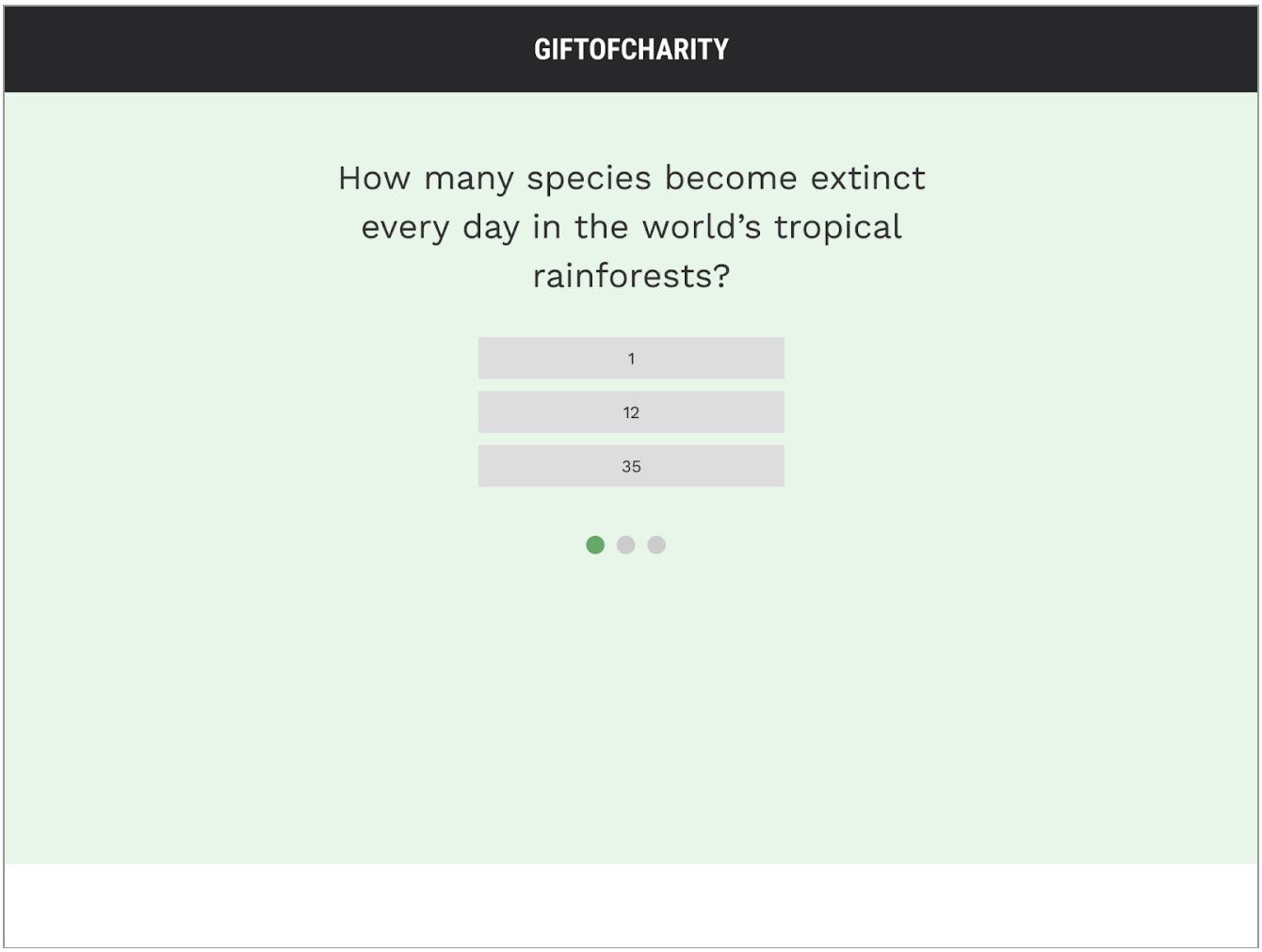


Imagen 16. Vista del cuestionario - Pregunta.

Una vez un donante realiza un donación y el receptor recibe la carta que contiene el enlace al regalo. Éste puede acceder a un cuestionario como en la imagen 16, donde podemos ver una pregunta del cuestionario asociado al proyecto de caridad el cual se ha donado. También se pueden ver tres posibles opciones y un indicador con forma de tres círculos que hacen referencia a la pregunta en la que se está actualmente.

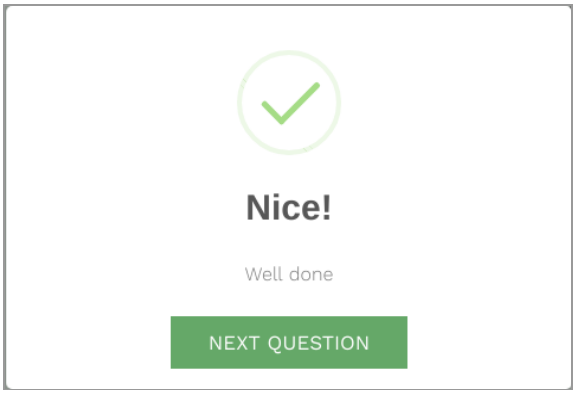


Imagen 17. Vista del Cuestionario - Mensaje de acierto.

Si al pulsar una respuesta ésta es correcta, se muestra un mensaje de acierto tal y como enseña la imágen 17. Y de forma simétrica, se lanza un mensaje de fallo en caso de que una respuesta no sea correcta.

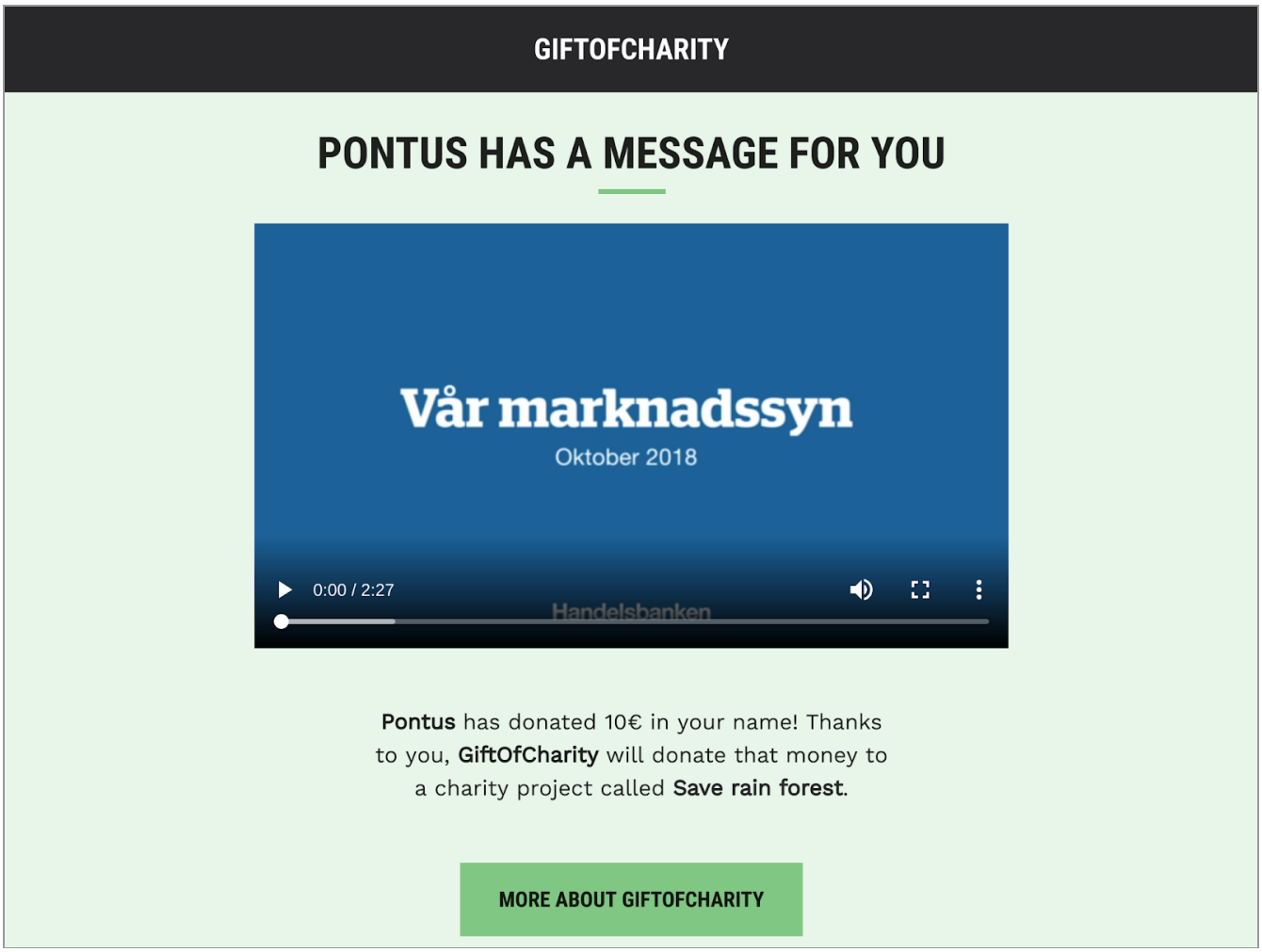


Imagen 18. Vista del video-mensaje.

En la imágen 18 vemos la vista que contiene el video-mensaje para el receptor del regalo. Contiene un visualizador de vídeo, además de un mensaje con el nombre del donante, la cantidad donada y un enlace al proyecto de caridad donado que lleva hacia la vista del proyecto de caridad asociado.

Finalmente hay un botón que redirige a la vista principal de la aplicación, de manera que los receptores puedan leer en la vista principal de qué va la aplicación web.

- Vistas de los e-mails:

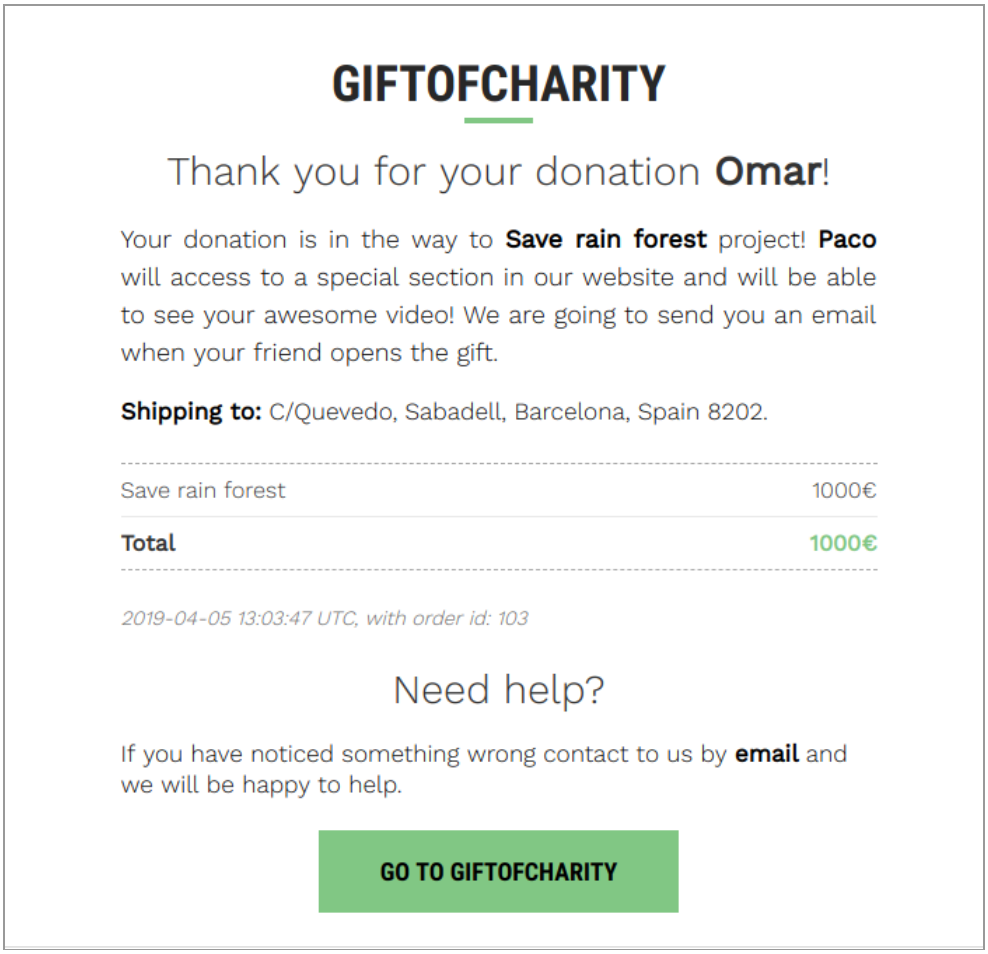


Imagen 19. E-mail - Recibo.

En la imágen 19 se puede ver el mensaje de e-mail que reciben los donantes un vez acaban el proceso de donación. Éste mensaje es un recibo e indica información del proceso de donación.

Tiene además un botón que redirige a la aplicación web.

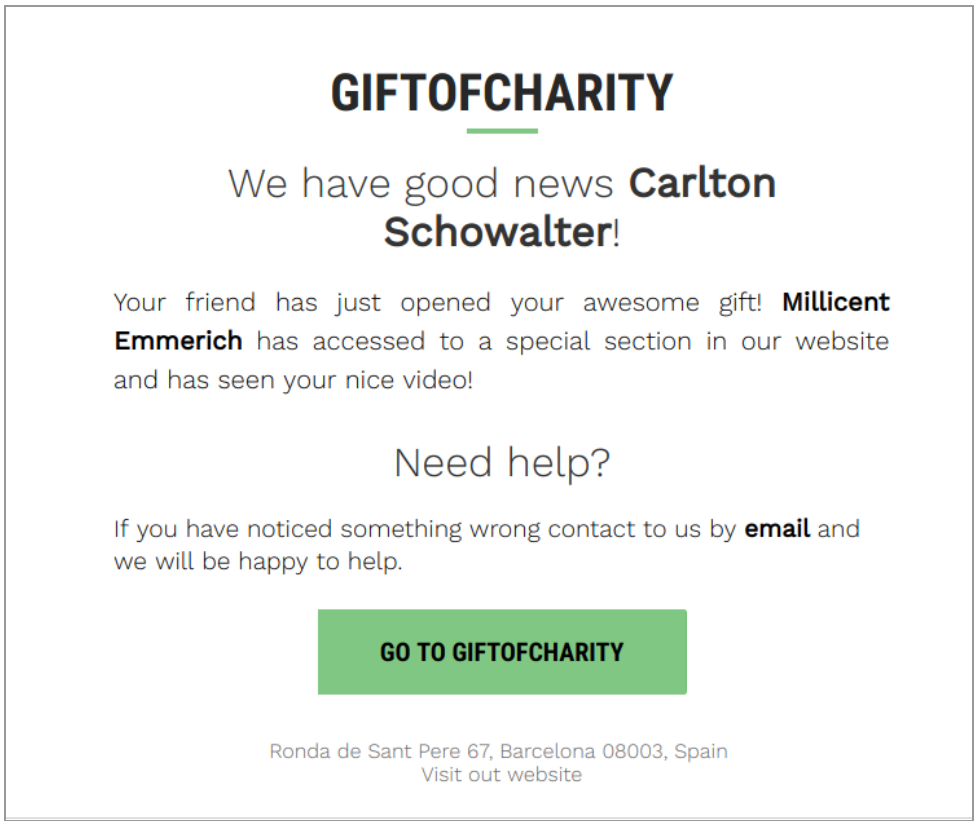


Imagen 20. E-mail - Notificación regalo abierto.

Como podemos ver en la imágen 20, se muestra el e-mail que recibe un donante una vez el receptor del regalo ha accedido al enlace que le lleva a su regalo.

La estética es similar al e-mail de los recibos y contiene un botón idéntico para redirigir al usuario a la aplicación web.

- Vista de la carta:

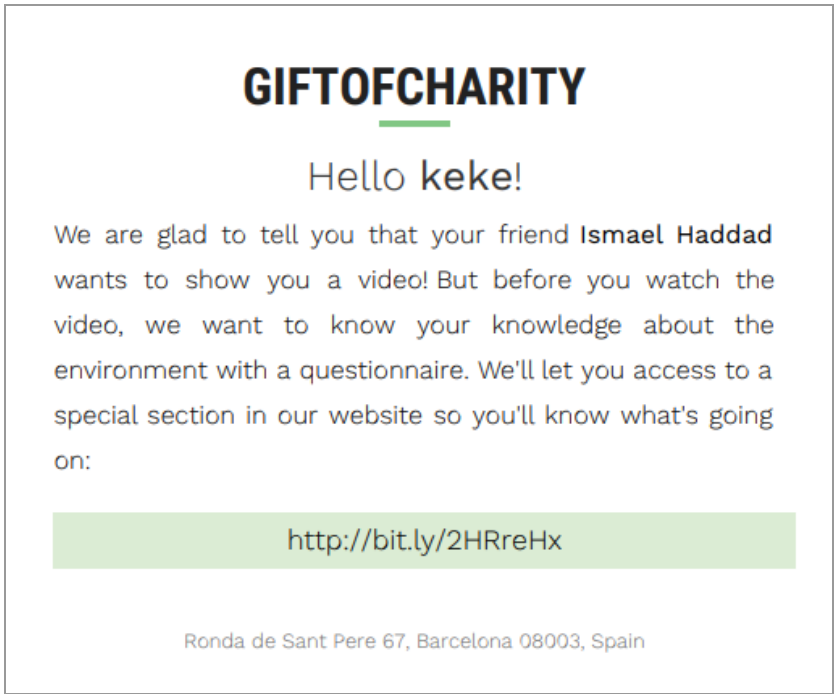


Imagen 21. Carta.

En la imagen 21 se ve la carta que reciben los receptores de regalos. Contiene un mensaje invitando al receptor de regalo a acceder al enlace que se muestra resaltado en la carta.

Panel de administración

- Vistas de los administradores:

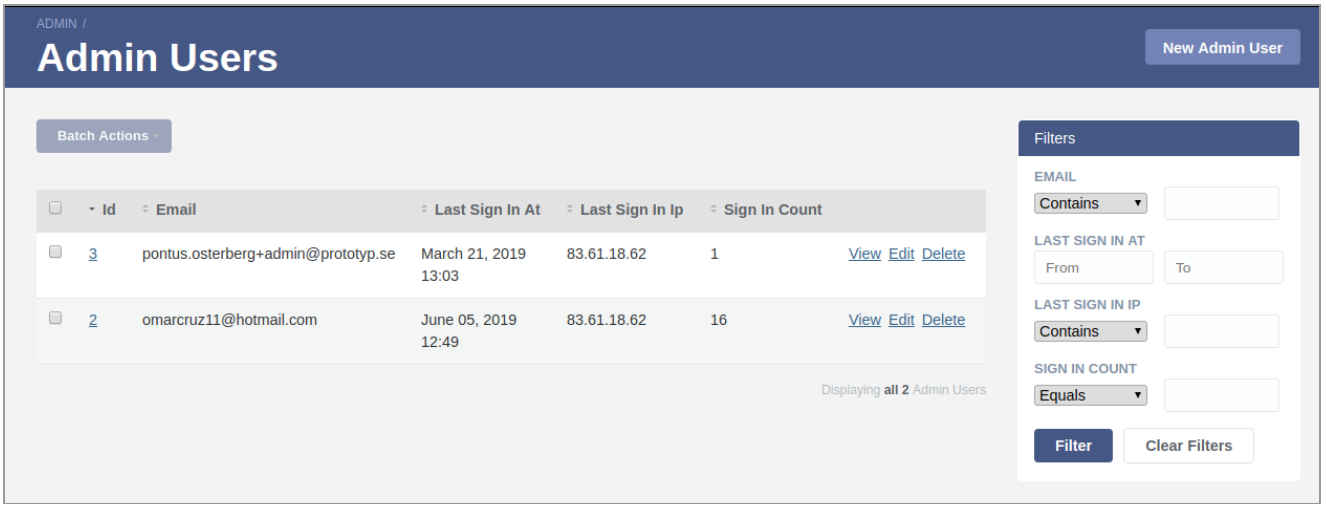


Imagen 22. Vista de la lista de administradores.

Desde el panel de administración se pueden ver todos los administradores del sistema. Con información básica de rastreo como la última vez que un administrador ha accedido al panel de

administración. Al lado derecho vemos un apartado de filtraje, desde donde se pueden filtrar entradas según condiciones. A su vez, cada entrada posee de 3 enlaces, uno para ver toda la información del administrador, otro que redirige a una vista para editar el email y contraseña y un botón de borrar, que borra al administrador.

Imagen 23. Vista de la creación de un administrador.

La vista de la imagen 23 se abre al pulsar el botón para crear administradores de la vista anterior (imagen 22). Desde aquí se puede introducir el e-mail del nuevo administrador, al que le llegará un mensaje con un enlace a la vista de resetear contraseña.

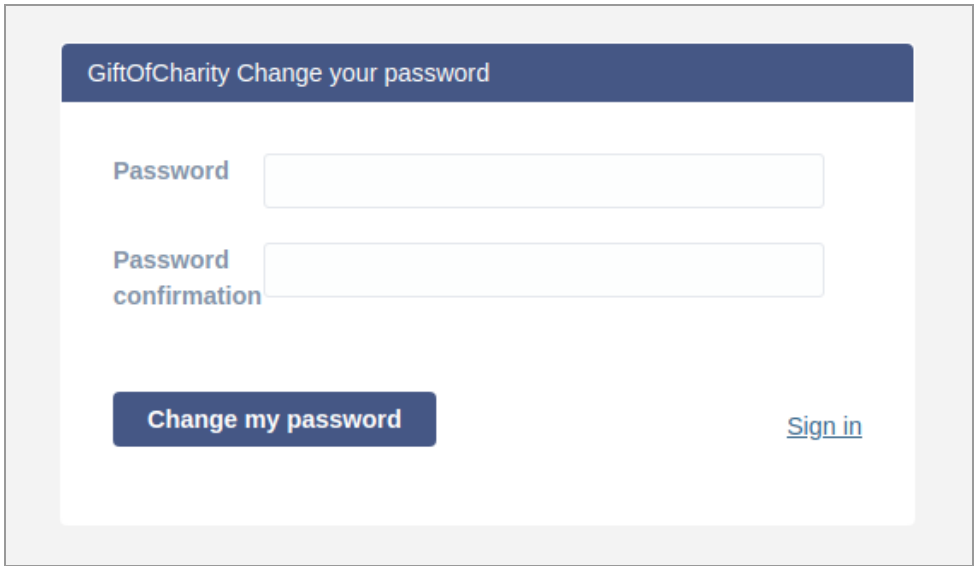


Imagen 24. Vista de resetear contraseña de un administrador.

Los nuevos administradores acceden a un enlace único desde el cual pueden introducir una contraseña para su nueva cuenta, tal y como se ve en la imagen 24.

- Vistas de los proyectos de caridad:

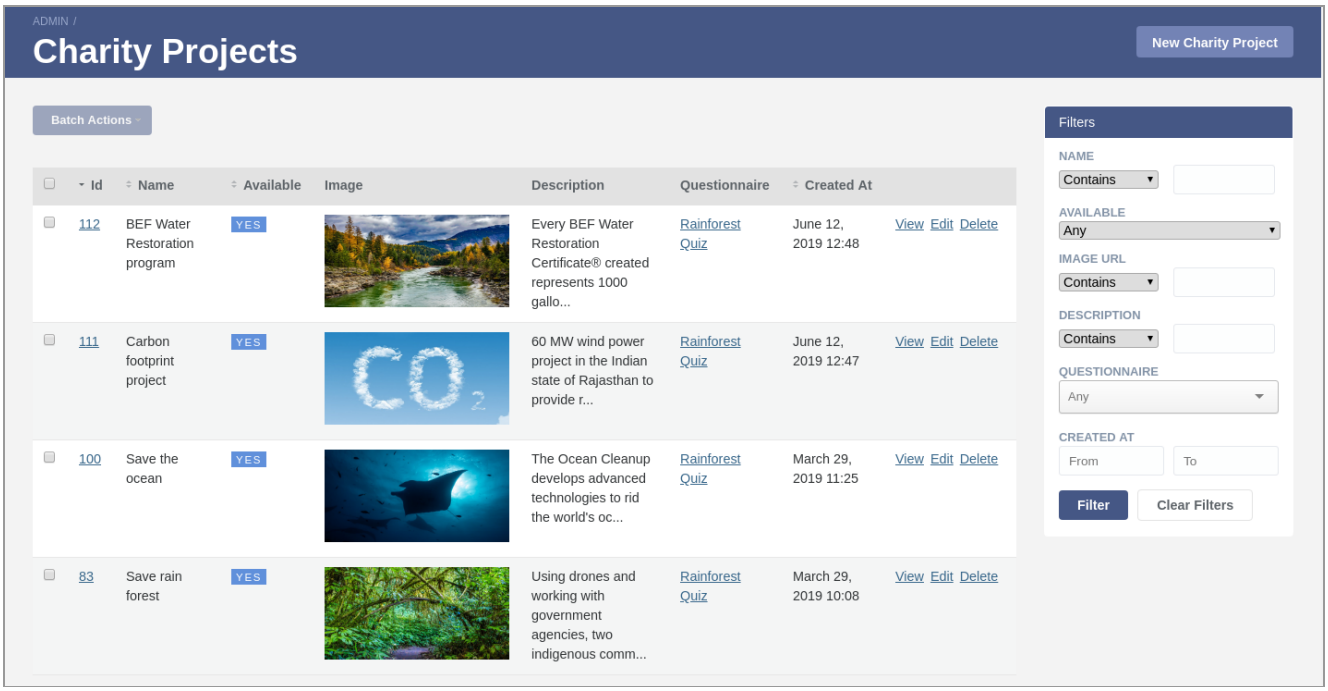


Imagen 25. Vista de la lista de proyectos de caridad.

En la imagen 25 podemos ver la lista de los proyectos de caridad. Cada entrada indica información básica como el nombre, la disponibilidad y el cuestionario relacionado. Además, también hay unos botones que sirven para ver toda la información del proyecto de caridad, editar y borrar.

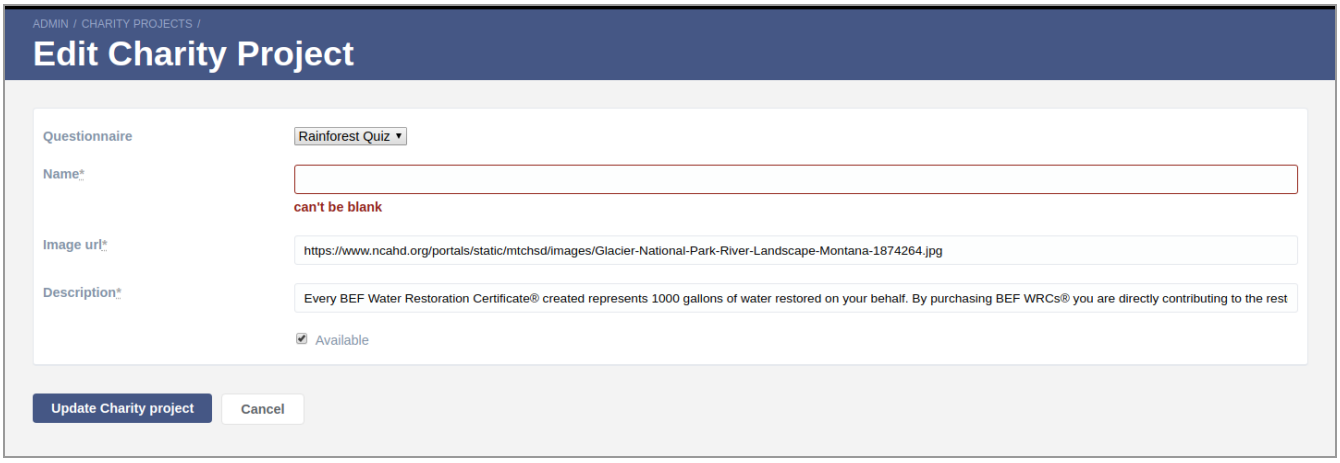


Imagen 26. Vista de edición de un proyecto de caridad.

En pulsar el botón de editar de la lista de proyecto de caridad, se nos abre una nueva vista tal y como vemos en la imagen 26. En esta vista se puede editar el nombre, el enlace de la imagen, la descripción del proyecto y seleccionar el cuestionario relacionado. Además, los formularios de edición muestran los errores introducidos en los campos (todas las vistas de edición).

Vistas de donaciones:

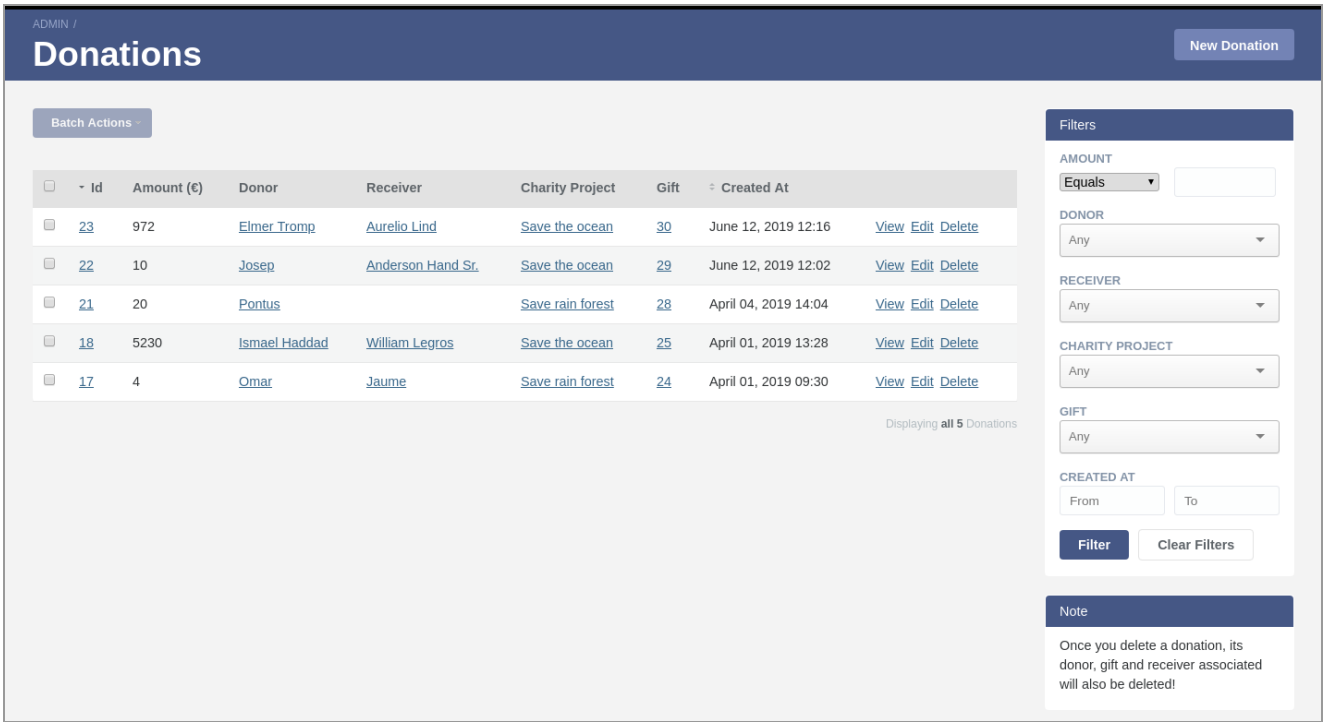


Imagen 27. Vista de la lista de donaciones

La imagen 27 muestra la vista de la lista de donaciones. Cada entrada de la lista muestra una relación con el donante, receptor, proyecto de caridad y regalo. Cuando se accede al enlace de estas relaciones, la vista redirige a otra que contiene información específica de las instancias relacionadas.

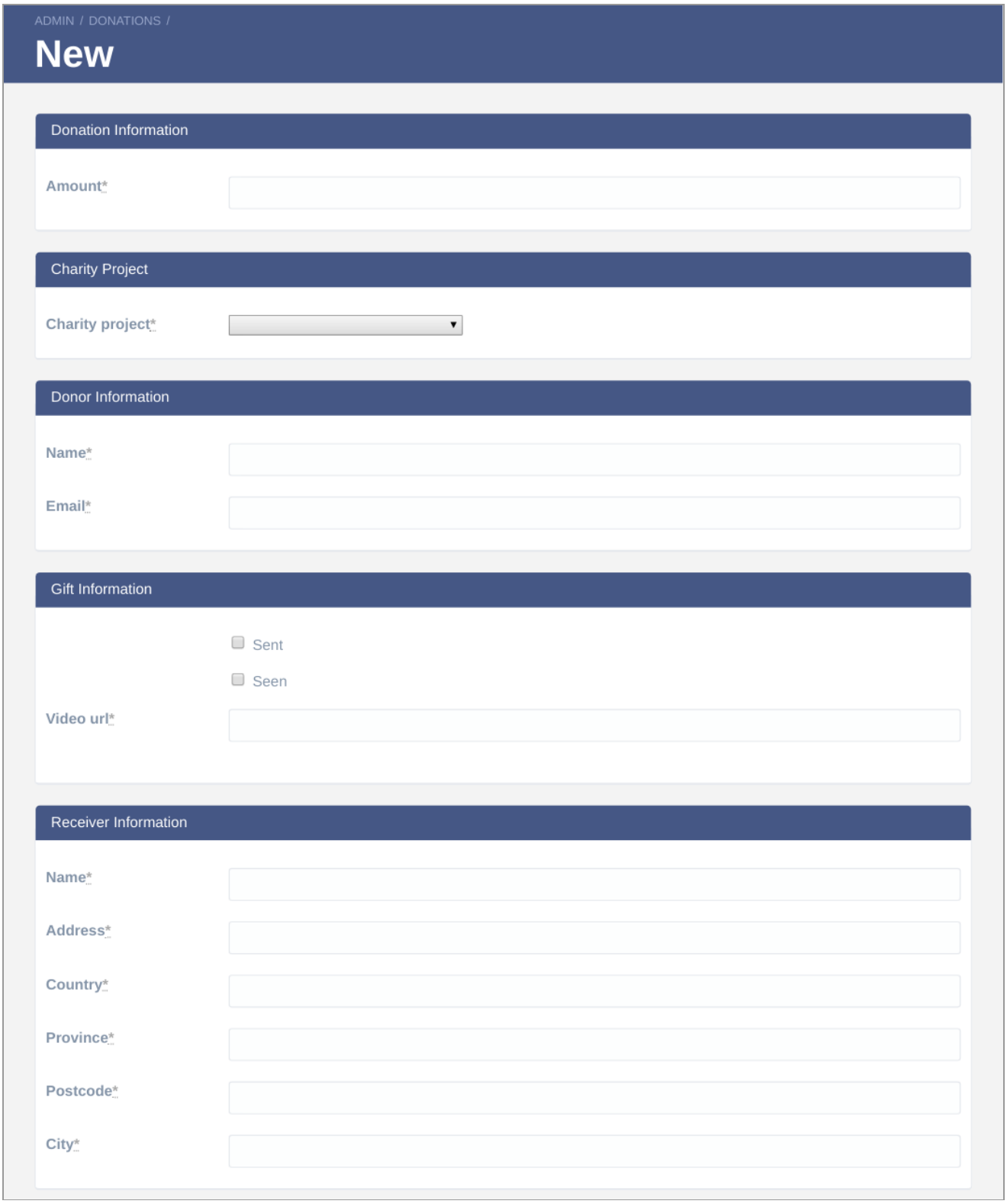


Imagen 28. Vista de la creación de una donación.

En esta vista de la imagen 28, se encuentra un formulario con todos los campos necesarios para crear una donación de forma unificada.

Imagen 29. Vista de la edición de una donación.

Respecto a las donaciones, se pueden modificar los campos como la cantidad de dinero donada o el proyecto de caridad asociado tal y como vemos en la imagen 29. Sin embargo, para editar la información del donante, regalo o receptor, se tendrá que ir a otra vista donde sólo se puedan editar los campos del modelo escogido.

- Vistas de los regalos:

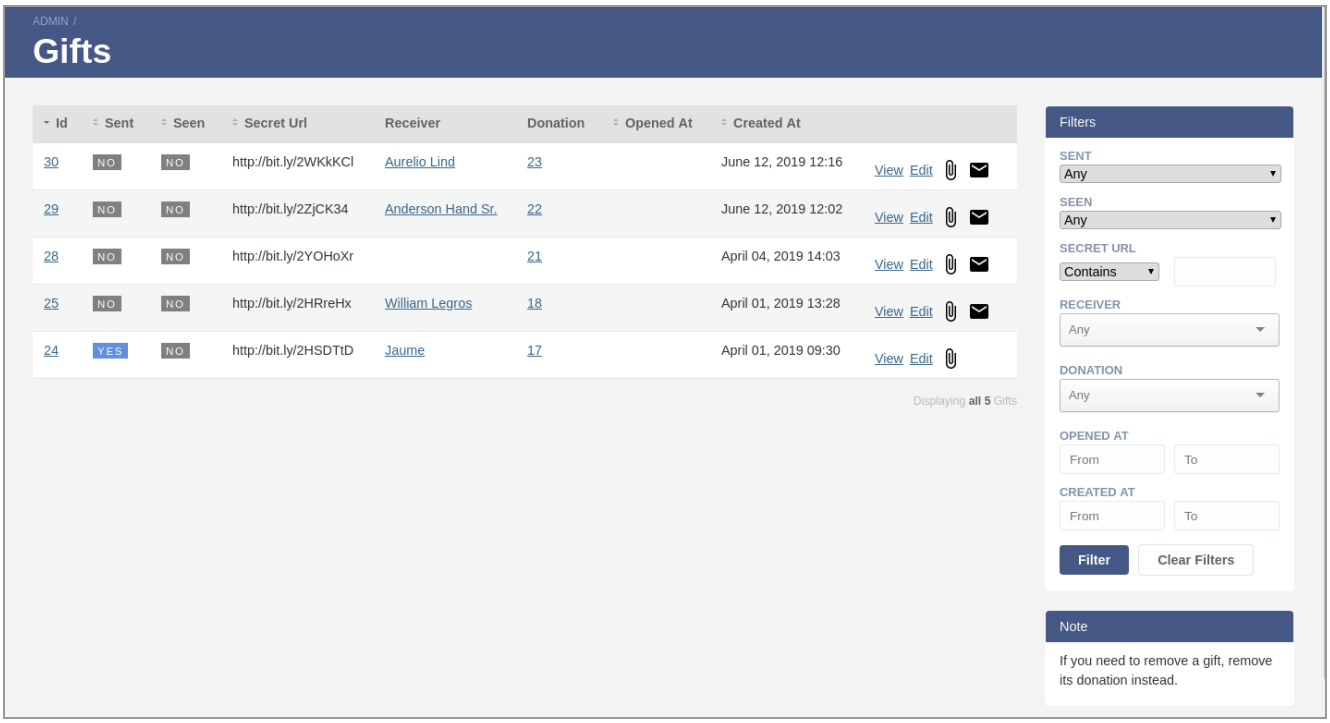


Imagen 30. Vista de la lista de regalos.

En la imagen 30 vemos la lista de regalos. Cada entrada de la lista contiene información respecto al estado del regalo (enviado y/o visto por el receptor), el enlace secreto al regalo, el receptor, la donación asociada, la fecha en la que se ha abierto el regalo y la fecha de creación. También, en el lado derecho podemos ver unos iconos para cada entrada que corresponden a las acciones de descargar la carta del regalo en pdf y marcar regalo como enviado.

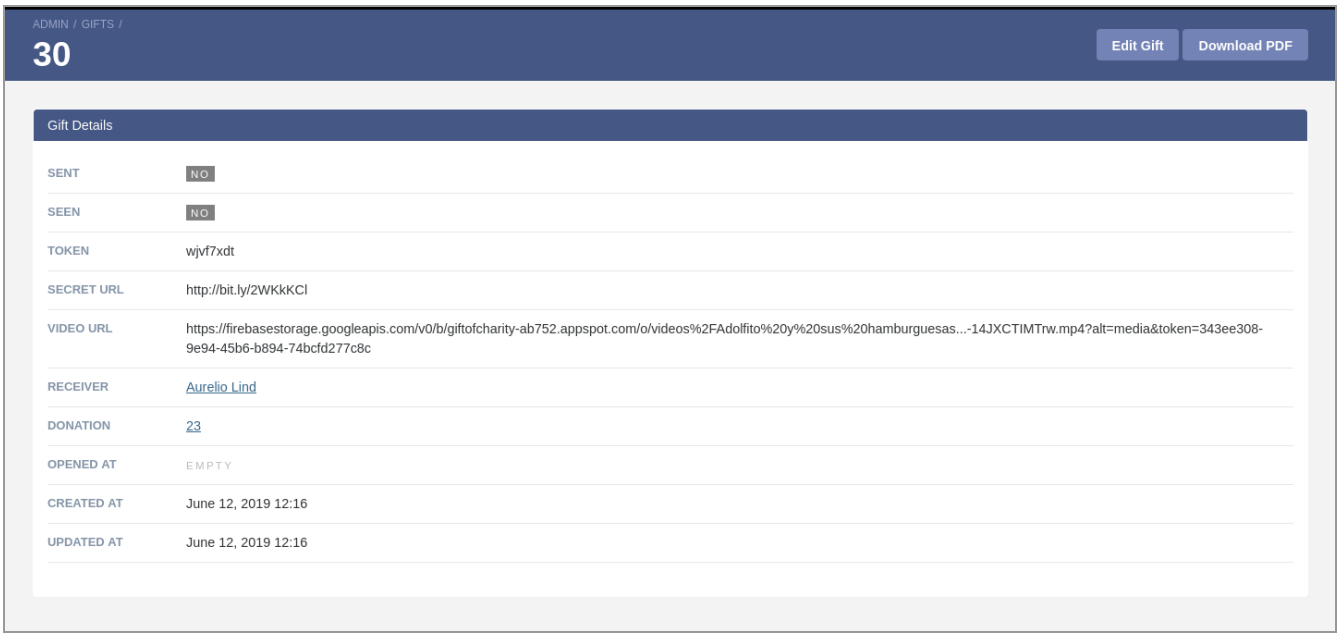


Imagen 31. Vista de la información de un regalo.

Al presionar sobre el enlace ‘view’ en cada entrada de la imagen 30, podemos acceder a una vista que cuenta con toda la información almacenada en la base de datos para cada regalo. De la misma manera, si recordamos la imagen 27 donde se muestra la lista de donaciones, cada entrada está relacionada con un regalo, que al presionar sobre el enlace, se muestra esta misma vista.

- Vistas de receptores y donantes:

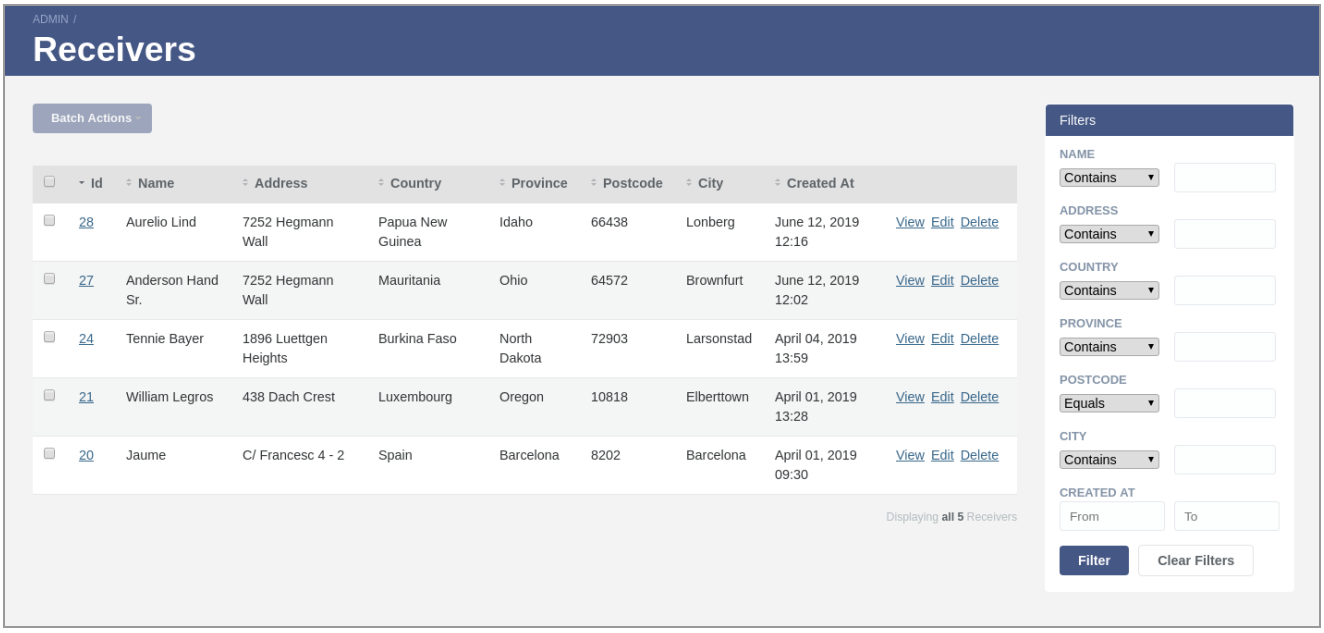


Imagen 32. Vista de la lista de receptores.

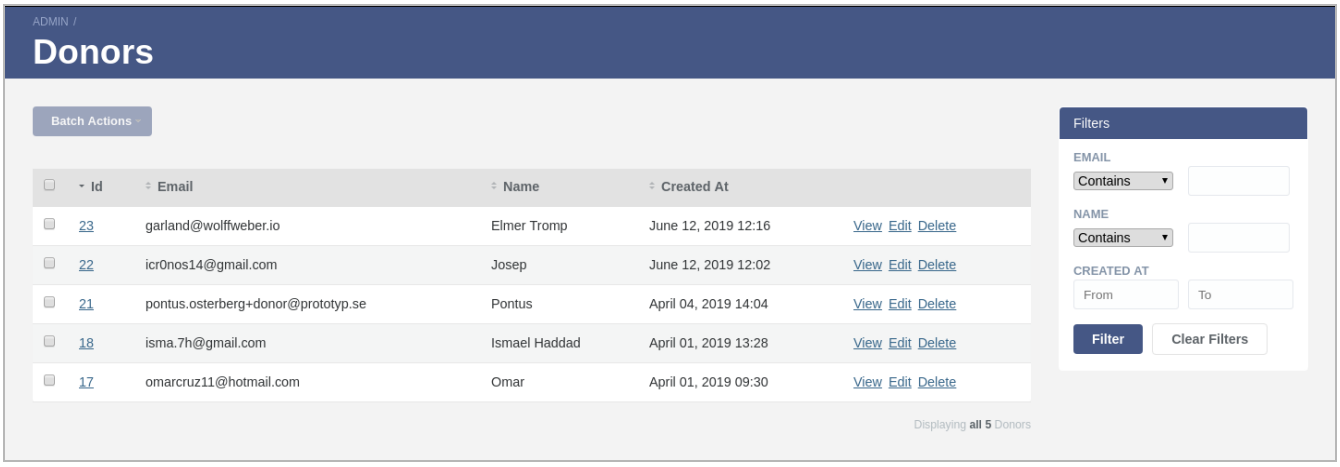


Imagen 33. Vista de la lista de donantes.

En las imágenes 32 e 33 podemos ver listas de receptores y donantes con su información. Cada una de las listas posee la posibilidad de ver en detalle todos los campos, editar o borrar cada entrada.

- Vistas de cuestionarios:

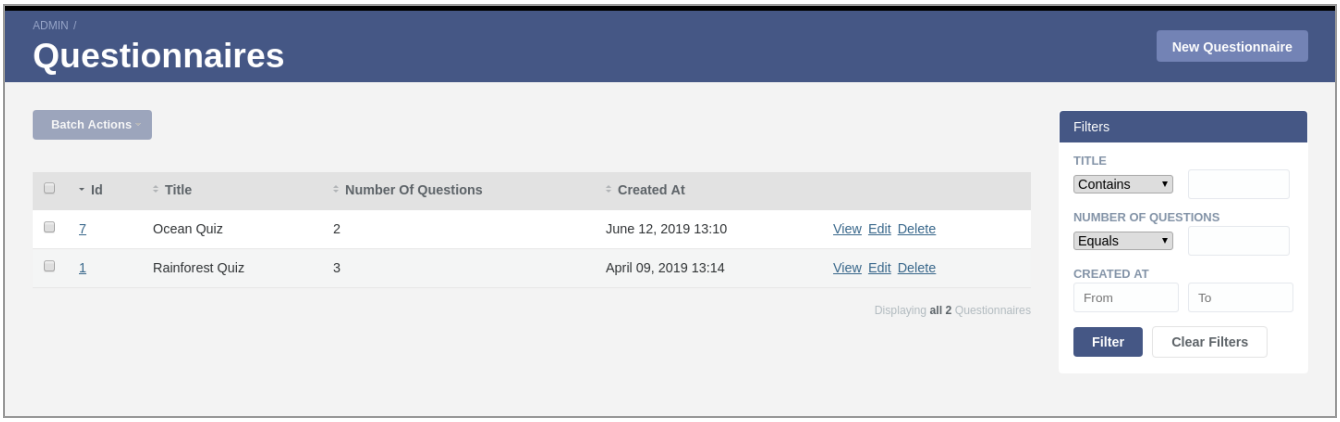


Imagen 34. Vista de la lista de cuestionarios.

En panel de administrador también cuenta con una lista de cuestionarios en la que se pueden los cuestionarios almacenados en el sistema. Al igual que las otras vistas de listados, para cada entrada contiene unas acciones que són ver en detalle la entrada, editar o borrar el cuestionario.

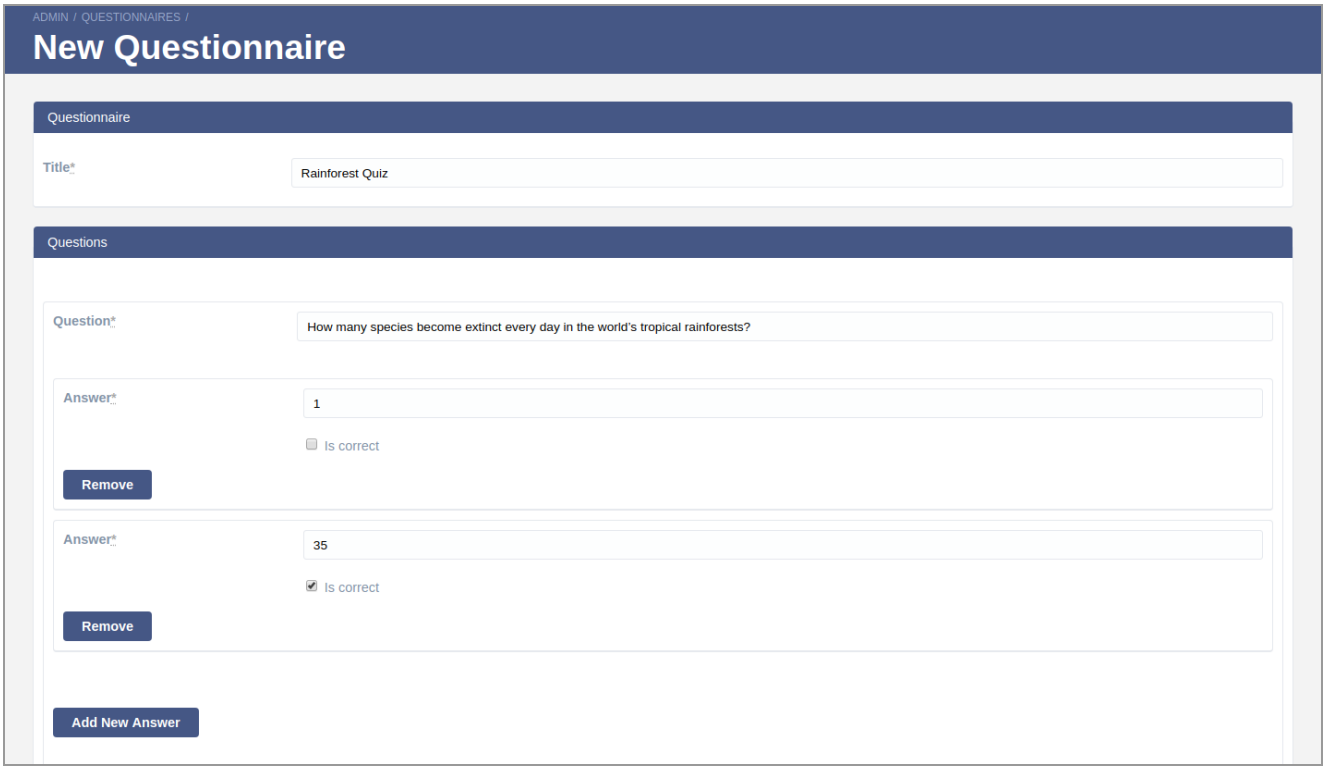


Imagen 35. Vista de creación de un cuestionario.

La imagen 35 muestra la creación de un cuestionario. En esta vista se pueden ir anidando las preguntas y respuestas que contendrá el cuestionario.

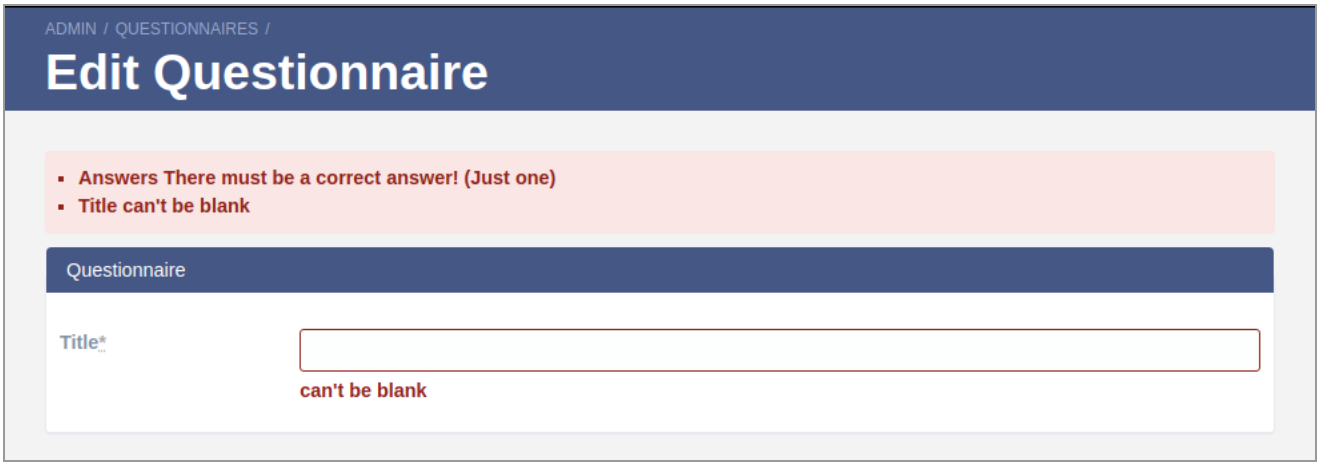


Imagen 36. Vista de creación de un cuestionario.

La creación o edición de cuestionarios cuenta con avisos de errores que indican el campo erróneo y el por qué. En la imagen 36 podemos ver los errores mostrados en dejar un cambio vación y en introducir una pregunta del cuestionario que no contiene ninguna respuesta correcta.

8.3. Capa de dominio

En esta capa se realizarán las llamada a todos los servicios de terceros que utilice la aplicación, a través del protocolo HTTP antes dicho. En esta capa también almacenaremos los modelos del sistema y permitiremos una vía de comunicación con la capa de presentación.

Al igual que para la capa de presentación hay herramientas que te permiten centrarte únicamente en la interfaz gráfica, hay otro tipo de herramientas llamados framework MVC que si bien también te permiten ocuparte de la interfaz gráfica, actualmente se suele usar más para la lógica y para la comunicación a la base de datos.

8.3.1. Diagrama de clases de diseño

Partiendo del esquema conceptual de los datos del capítulo de especificación, he realizado unos cambios con tal de obtener el diagrama de clases de diseño. Este paso es necesario pues en esta etapa de diseño hay una limitación tecnológica y no se pueden implementar todos los conceptos usados en la especificación [21].

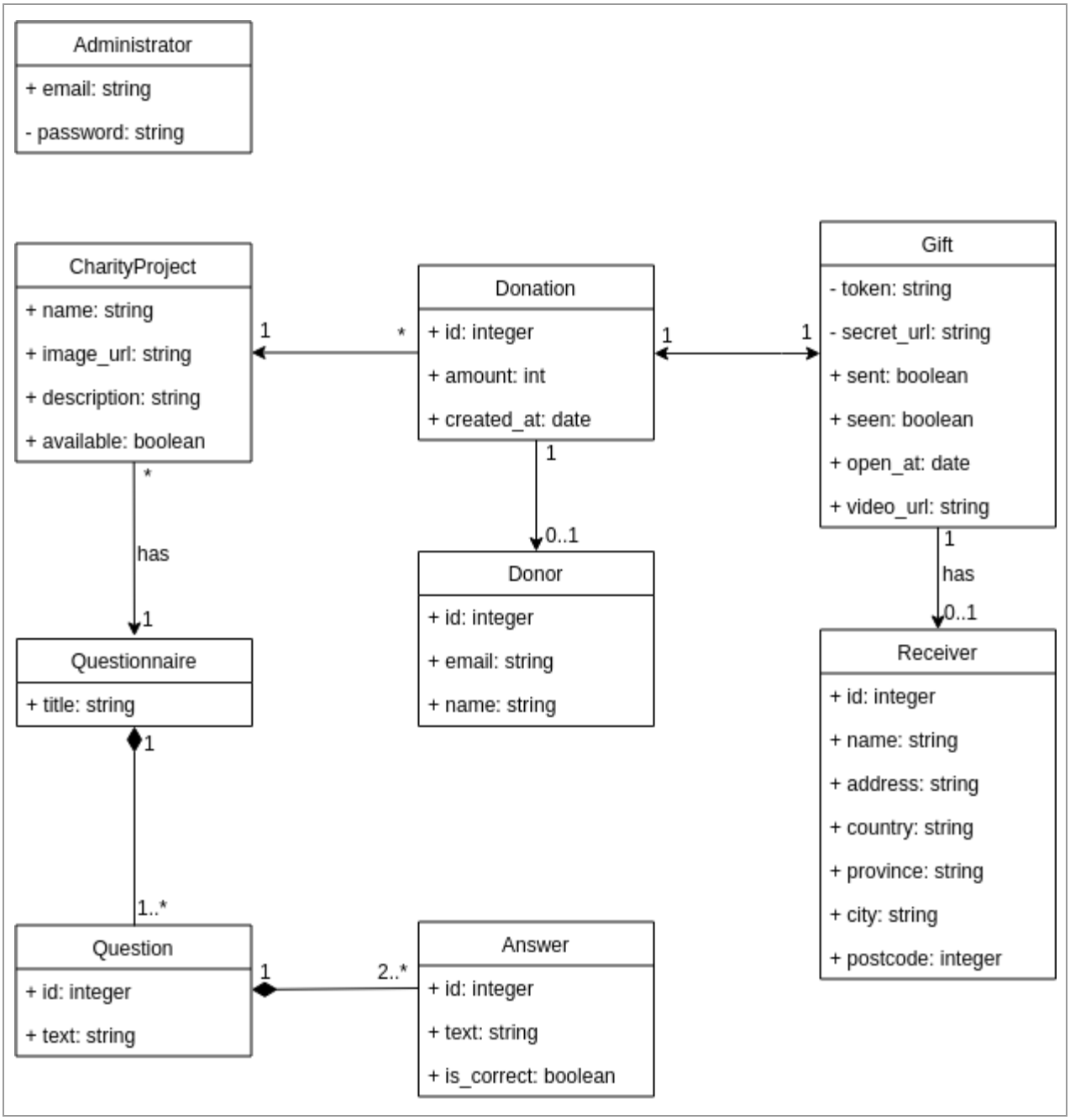


Imagen 37. Diagrama de clases de diseño.

Restricciones textuales

1. Claves externas: (Administrator, email), (CharityProject, name), (Questionnaire, title), (Question, id), (Answer, id), (Donor, id), (Gift, token), (Receiver, id), (Donation, id).
2. Un regalo no puede estar marcado como 'visto' si no ha sido enviado a su receptor.
3. La fecha de visualización de un regalo ha de ser posterior a la fecha de la creación de la donación a la que pertenece.

4. No pueden haber dos respuestas marcadas como correctas que pertenezcan a la misma pregunta.
5. No pueden haber dos respuestas iguales que pertenezcan a la misma pregunta.
6. No pueden haber dos preguntas iguales que pertenezcan al mismo cuestionario.

Cambios realizados respecto al esquema conceptual de los datos

- Se ha añadido la navegabilidad entre clases. De esta manera, podemos ver si podemos atravesar una asociación de una clase hacia otra.

- Eliminación de la clase vídeo. Debido a que esta clase solo tiene un atributo (url) y la relación era de uno a uno, se ha añadido el atributo video_url a la clase regalo (gift).

- Se ha añadido la visibilidad a cada atributo de las clases. De esta manera podemos indicar si las otras clases pueden acceder a estos atributos o no. La visibilidad se indica mediante un símbolo antes del atributo y puede ser pública (+) si cualquier clase que ve la clase puede acceder. O privada (-) si solo la clase puede ver el atributo.

8.3.2. API REST

Tal y como se ha comentado en el apartado de infraestructura, la comunicación entre el servidor front-end y servidor back-end se realiza mediante el protocolo HTTP. Se ha construido una API, de manera que la comunicación entre los clientes (navegadores web) y el servidor se haga de forma estándar. Además, se hará uso de unas restricciones definidas por la tesis de Roy Fielding, padre de la especificación HTTP, donde se haya el término REST. [22]

El hecho de aplicar estas restricciones ha hecho que se cree una API REST. Pondré algunas de las restricciones aplicadas [23]:

- Protocolo cliente/servidor sin estado: Todas las operaciones contienen la información necesaria para ser ejecutadas. No se necesita un estado previo, ni para el cliente, ni para el servidor.
- Cuatro operaciones: Se utilizarán las operaciones POST para crear, GET para leer y consultar, PUT para editar y DELETE para borrar.
- Objetos en REST manipulados con URI: La URI (que es el identificador del objeto que se quiere tratar, situado en la url de la petición) facilitará el acceso a la información.

Funcionamiento

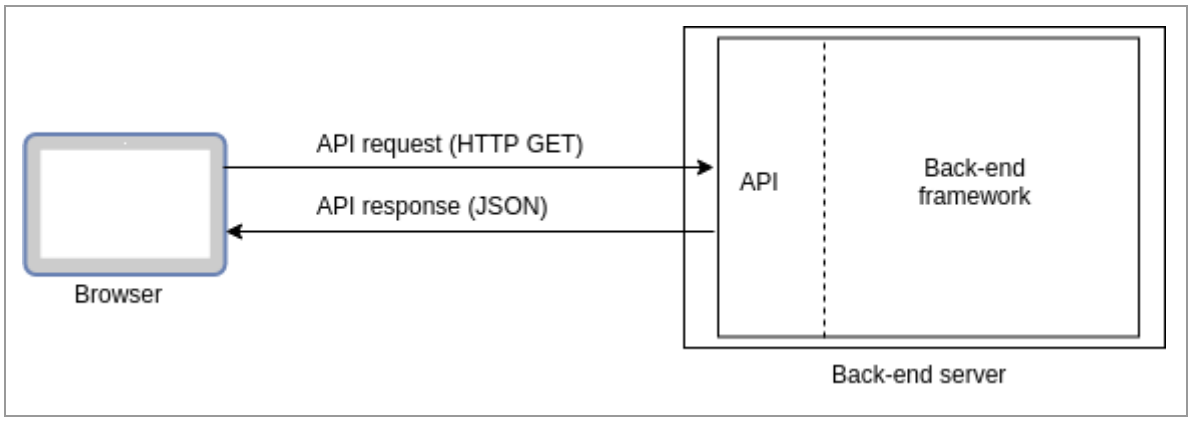


Imagen 38. Operación GET mediante el protocolo HTTP.

Cada una de las operaciones del cliente al servidor back-end estará dividida en petición y respuesta tal y como vemos en la imagen 38. De forma general, podemos decir que la petición consta de un método, url y parámetros que se ha de mandar al servidor, mientras que la respuesta estará en notación JSON, que es un formato de texto sencillo e intuitivo utilizado para el intercambio de datos donde los datos se representan mediante clave y valor, como en la siguiente imagen:


```
[
  {
    id: 4,
    email: "jenettedenesik@gerlach.io",
    name: "Scott Fadel",
    created_at: "2019-04-23T09:27:08.882Z",
    updated_at: "2019-04-23T09:27:08.882Z"
  },
  {
    id: 3,
    email: "dusty@lebsack.io",
    name: "Jalisa Greenholt II",
    created_at: "2019-04-23T09:27:08.853Z",
    updated_at: "2019-04-23T09:27:08.853Z"
  },
  {
    id: 2,
    email: "ulalarkin@luettgen.com",
    name: "Ms. Shiela Gulgowski",
    created_at: "2019-04-23T09:27:08.824Z",
    updated_at: "2019-04-23T09:27:08.824Z"
  },
  {
    id: 1,
    email: "lora@aufderhar.biz",
    name: "Polly Paucek",
    created_at: "2019-04-23T09:27:08.787Z",
    updated_at: "2019-04-23T09:27:08.787Z"
  }
]
```


Imagen 39. Ejemplo de una respuesta de una API en formato JSON.

Rutas

Una vez explicado el funcionamiento, pasamos a nombrar las operaciones que tiene nuestra aplicación y que se consume en el lado del cliente (las operaciones que tendrán el panel de administrador no se especificarán, pues han sido generadas a través de una librería).

Épica	Método	Ruta	Descripción
Proyectos de caridad	GET	/charity_projects	Devuelve todos los proyectos de caridad disponibles del sistema.
Proyectos de caridad	GET	/charity_projects/{id}	Devuelve el proyecto de caridad con la id {id}.
Donaciones	POST	/donations	Realiza el proceso de donación a partir de la información obtenida a través del proceso de donación.
Regalos	GET	/gifts	Devuelve la información de un regalo a partir de un código llamado 'token' que se ha de pasar como parámetro.
Cuestionarios	GET	/questionnaires/{id}	Devuelvo la información (preguntas y respuestas incluidas) de un cuestionario a partir de una id {id}.

Tabla 48. Rutas de la API del servidor back-end.

8.3.3. Controladores

La letra C en un framework MVC como Ruby on Rails hace referencia a los controladores. Los controladores aceptan peticiones en el servidor donde esté alojado el proyecto y son los responsables de producir una respuesta correcta a los clientes [24]. Afortunadamente, Ruby on Rails proporciona una gran capa de abstracción y convenciones, por lo que se pueden escribir controladores de manera rápida y fácil.

Si recordamos el apartado anterior, la comunicación entre el cliente y el servidor back-end se hace a través de peticiones HTTP y es aquí en los controladores donde se desarrolla la lógica de cada una de las rutas de la API.

En el desarrollo de éste proyecto se ha hecho uso principalmente de cuatro controladores, donde cada unos de estos controladores tiene acciones que contienen la lógica de cada una de las rutas de la API del servidor back-end. Los controladores que tenemos son:

1. Proyectos de caridad (charity_projects): Contiene las acciones: obtener proyectos de caridad y obtener un proyecto de caridad, que corresponden a las historias de usuario: *Listado de proyectos de caridad* e *Información de un proyecto de caridad* respectivamente.

2. Donaciones (donations): Contiene la acción: realizar donación, que corresponde a la historia de usuario: *Realizar donación*.
3. Regalos (gifts): Contiene la acción: abrir regalo, que corresponde a la historia de usuario: *Acceso al regalo*.
4. Cuestionarios (questionnaires): Contiene la acción: obtener cuestionario, que corresponde a la historia de usuario: *Consumir cuestionario*.

Sin embargo, además de éstos controladores, hay otros que están puestos de forma implícita debido al panel de administración explicado más adelante. En la imagen de abajo vemos los controladores antes comentados y además, en el controlador de regalo vemos la acción 'downloadPDF', que está definida para que el panel de administrador haga uso de él.

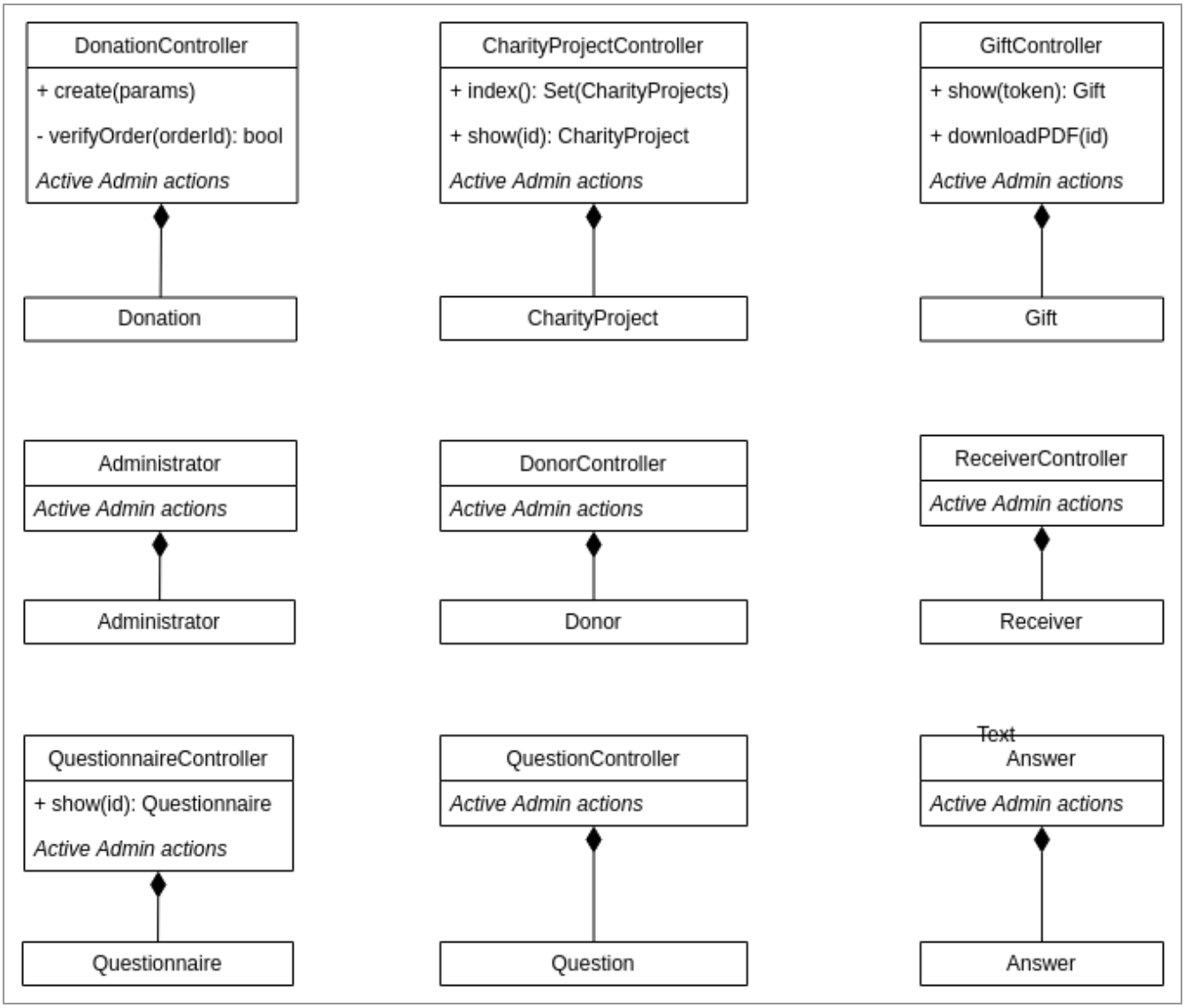


Imagen 40. Controladores del proyecto con sus operaciones.

API

Cada una de las acciones de los controladores se mapea a una ruta y esta ruta está especificada brevemente en la tabla 48 del apartado API REST, por lo que en este subapartado he especificado las rutas con mayor profundidad dentro de una tabla.

- Obtener proyectos de caridad**:** Devuelve todos los proyectos de caridad disponibles del sistema.

Petición	
Método	URL
GET	/charity_projects
Respuesta	
Estado	Respuesta
200	La respuesta es un objeto conteniendo una lista de proyectos de caridad (array) con información básica. Un ejemplo de respuesta es: `[{ "id": 1, "name": "Save rain forest", "imageUrl": "https://images.pexels.com/photos/904807/pexels-photo-904807.jpeg?cs=srgb&dl=branches-daylight-environment-904807.jpg&fm=jpg" }, { "id": 2, "name": "Save the ocean", "imageUrl": "https://img.depresident.com/wp-content/uploads/2016/02/Save-the-Ocean-T-shirts.jpg" }]`
500	`{"message":"Something went wrong. Please try again later."}`

Tabla 49. Especificación de la ruta */charity_projects* (GET).

- Obtener un proyecto de caridad**:** Devuelve información del proyecto de caridad a partir de un identificador.

Petición

Método	URL		
GET	/charity_projects/{id}		
Parámetros			
Tipo	Parámetro	Valor	Descripción
URI	id	number	Identificador del proyecto de caridad del cual se quiere obtener información.
Respuesta			
Estado	Respuesta		
200	La respuesta es un objeto con información del proyecto de caridad que corresponde al identificador proporcionado. Un ejemplo de respuesta es: `{ "id": 83, "name": "Save rain forest", "imageUrl": "https://images.pexels.com/photos/904807/pexels-photo-904807.jpeg?cs=srgb&dl=branches-daylight-environment-904807.jpg&fm=jpg", "description": "Using drones and working with government agencies, two indigenous communities in Peru have gone from losing 5% of their land to 0% deforestation. Indigenous communities are protecting the rainforest even as they face pressure and violence from illegal coca growing, drug trafficking and logging.", "questionnaireId": 42 }`		
404	`{"message":"Couldn't find CharityProject"}`		
500	`{"message":"Something went wrong. Please try again later."}`		

Tabla 50. Especificación de la ruta */charity_projects/{id}* (GET).

- Realizar donación**:** Realiza el proceso de donación a partir de la información obtenida a través del proceso de donación.

Petición	
Método	URL
POST	/donations

Parámetros			
Tipo	Parámetro	Valor	Descripción
POST	orderId	string	Es el identificador de la compra. Se tendrá que verificar que la transacción es válida para poder insertar la donación en el sistema.
POST	videoUrl	string	Es el enlace que se ha generado para el vídeo que recibirá el receptor del regalo.
POST	address	string	Es la dirección donde se enviará el regalo.
POST	city	string	Indica la ciudad donde se enviará el regalo.
POST	country	string	Indica el país donde se enviará el regalo.
POST	province	string	Indica la provincia donde se enviará el regalo.
POST	postcode	number	Indica el código postal donde se enviará el regalo.
POST	email	string	Indica el e-mail del donante.
POST	itemId	number	Indica el identificador del proyecto de caridad.
POST	amount	number	Es la cantidad de dinero donada.
POST	donorName	string	Es el nombre del donante.
POST	receiverName	string	Es el nombre del receptor del regalo.

Respuesta	
Estado	Respuesta

200	El estado 200 indica que la donación fue realizada correctamente. La respuesta es un objeto con información de la donación producida. Un ejemplo de respuesta es:
	
	`{`
	`"id": 1,`
	`"amount": 10,`
	`"giftId": 1,`
	`"donorId": 1,`
	`"charityProjectId": 1,`
	`"createdAt": "2019-04-11T15:20:33.339Z"`
	`}`

402	{"message":"Payment required"}
500	{"message":"Something went wrong. Please try again later."}

Tabla 51. Especificación de la ruta */donations* (POST).

- Abrir regalo: Devuelve información de un regalo a partir de un token.

Petición			
Método	URL		
GET	/gifts		
Parámetros			
Tipo	Parámetro	Valor	Descripción
query	token	string	Es el token que se usará para verificar el regalo.
Respuesta			
Estado	Respuesta		

200	La respuesta es un objeto con información del regalo que corresponde al token proporcionado. Un ejemplo de respuesta es:
	
	`{`
	`"videoUrl":`
	`"https://firebasestorage.googleapis.com/v0/b/giftofcharity-ab752.appspot.com/o/videos%2FSHB_1851_03_Var_Marknadssyn_181003_webb_360p_0530c0b5.mp4?alt=media&token=fea268f1-8c17-4cf3-ae49-203d15b520bb",`
	`"donorName": "Pontus",`
	`"amount": 10,`
	`"charityProject": {`
	`"id": 83,`
	`"name": "Save rain forest",`
	`"imageUrl":`
	`"https://images.pexels.com/photos/904807/pexels-photo-904807.jpeg?cs=srgb&dl=branches-daylight-environment-904807.jpg&fm=jpg",`
	`"description": "Using drones and working with government agencies, two indigenous communities in Peru have gone from losing 5% of their land to 0% deforestation. Indigenous communities are protecting the rainforest even as they face pressure and violence from illegal coca growing, drug trafficking and logging.",`		

	<pre>"available": true,
	"questionnaireId": 1
	}
	}</pre>

400	{"message":"There is no gift with that token!"}
500	{"message":"Something went wrong. Please try again later."}

Tabla 52. Especificación de la ruta */gifts* (GET).

- Obtener cuestionario**:** Devuelvo la información (preguntas y respuestas incluidas) de un cuestionario a partir de un identificador.

Petición			
Método	URL		
GET	/questionnaires/{id}		
Parámetros			
Tipo	Parámetro	Valor	Descripción
URI	id	number	Identificador del cuestionario del cual se quiere obtener la información.
Respuesta			
Estado	Respuesta		

200	La respuesta es un objeto con información del cuestionario que corresponde al identificador proporcionado. Un ejemplo de respuesta es:
	
	<pre>{
	"title": "Rainforest Quiz",
	"questions": [
	{
	"number": 1,
	"prompt": "How many species become extinct every day in the world's tropical rainforests?",
	"answers": [
	"1",
	"12",
	"35"</pre>		

	`"answers": ["Plants", "Frogs", "Spiders", "Snakes"], "correct": { "index": 2 } }, { "number": 2, "prompt": "Traditional rainforest hunters use blow pipes and poison tipped darts. Where do they get the poison from?", "answers": ["Plants", "Frogs", "Spiders", "Snakes"], "correct": { "index": 1 } }, { "number": 3, "prompt": "Plants from the rainforest have helped us treat...", "answers": ["Leukaemia", "Breast cancer", "Asthma", "All of these"], "correct": { "index": 3 } }] }`
404	`{"message":"Couldn't find Questionnaire"}`
500	`{"message":"Something went wrong. Please try again later."}`

Tabla 53. Especificación de la ruta */questionnaires/{id}* (GET).

8.3.4. Servicios

Si recordamos la imagen 4 del apartado Infraestructura, tenemos un componente llamado servicios (services) en el diagrama. Como comenté este componente es un cúmulo de conexiones entre el

servidor back-end y diversos servicios ofrecidos por empresas distintas. En esta sección comentaré cada uno de los servicios se usan en la aplicación.

Paypal

En la historia de usuario *Realizar donación,* es necesario poder introducir un método de pago con tal de realizar la donación. Pero antes, ¿Cómo llega el dinero a los proyectos de caridad? Para tomar esta decisión, me basé en un complemento para tiendas online llamado: CO2ok [25]. Este complemento añade un botón de donación en las compras de productos online, y permite añadir una cantidad de dinero extra en el carrito de compra. Entonces, la empresa detrás de este complemento manda una factura cada tres meses a las tiendas online que usan este complemento con tal de destinar ese dinero para proyectos contra el dióxido de carbono. Entonces, la idea de GiftOfCharity es poder destinar todo el dinero a una cuenta común y gracias al registro de donaciones del panel de administrador, saber que cantidad repartir a los proyectos de caridad o incluso poder destinar más fondos a los que más importantes sean.

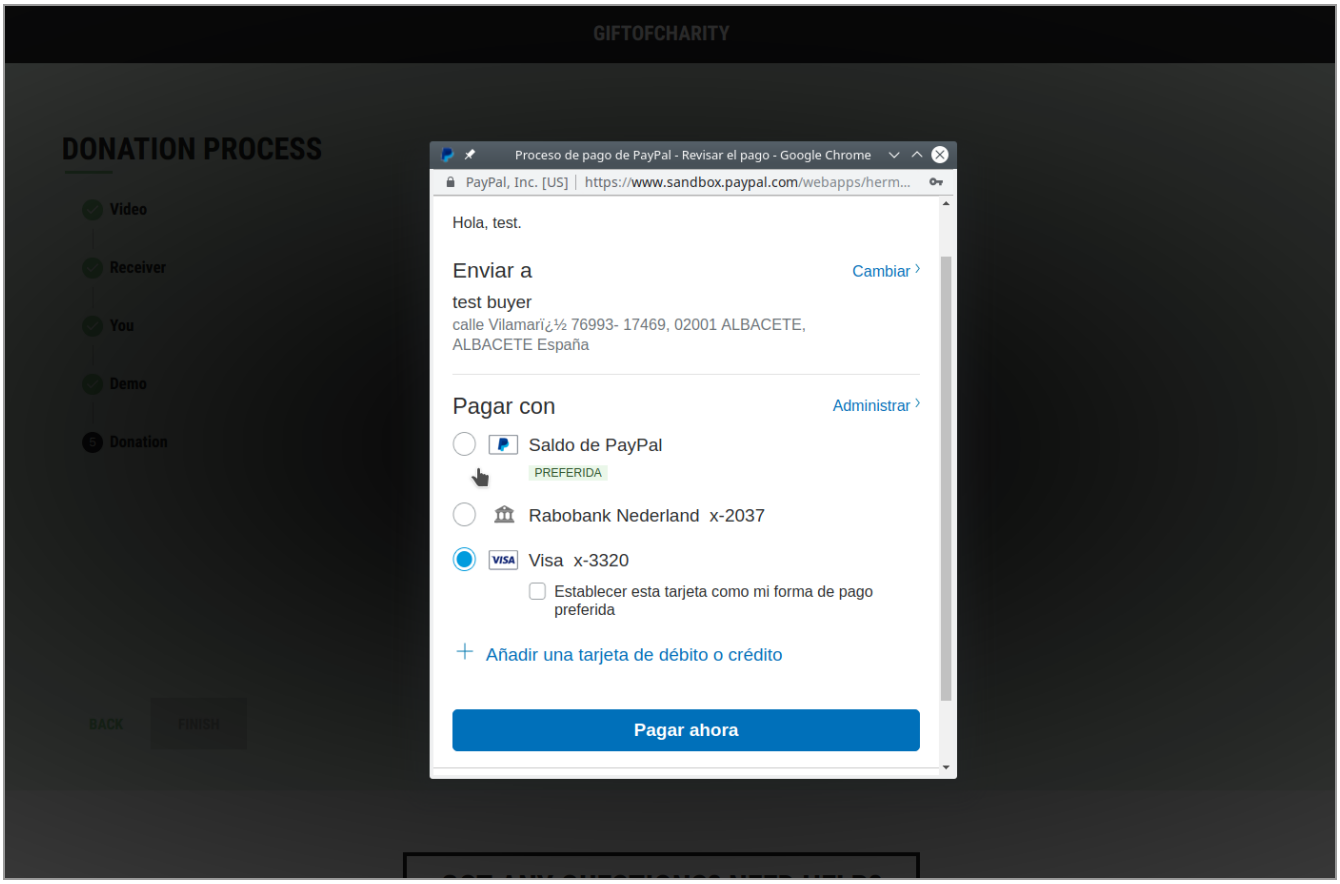


Imagen 41. Ventana de pago con Paypal integrada en GiftOfCharity.

Respecto a la tecnología para realizar los pagos, siempre me ha llamado la atención la sencillez de Paypal. A través de un email y contraseña puedes realizar pagos de forma muy fácil, y además, permite también realizar pagos con tarjeta de crédito sin la necesidad de crearse una cuenta. Además, pregunte a mis compañeros de la oficina si conocían algún servicio fácil y fiable y todos coincidieron en que Paypal era la opción indicada.

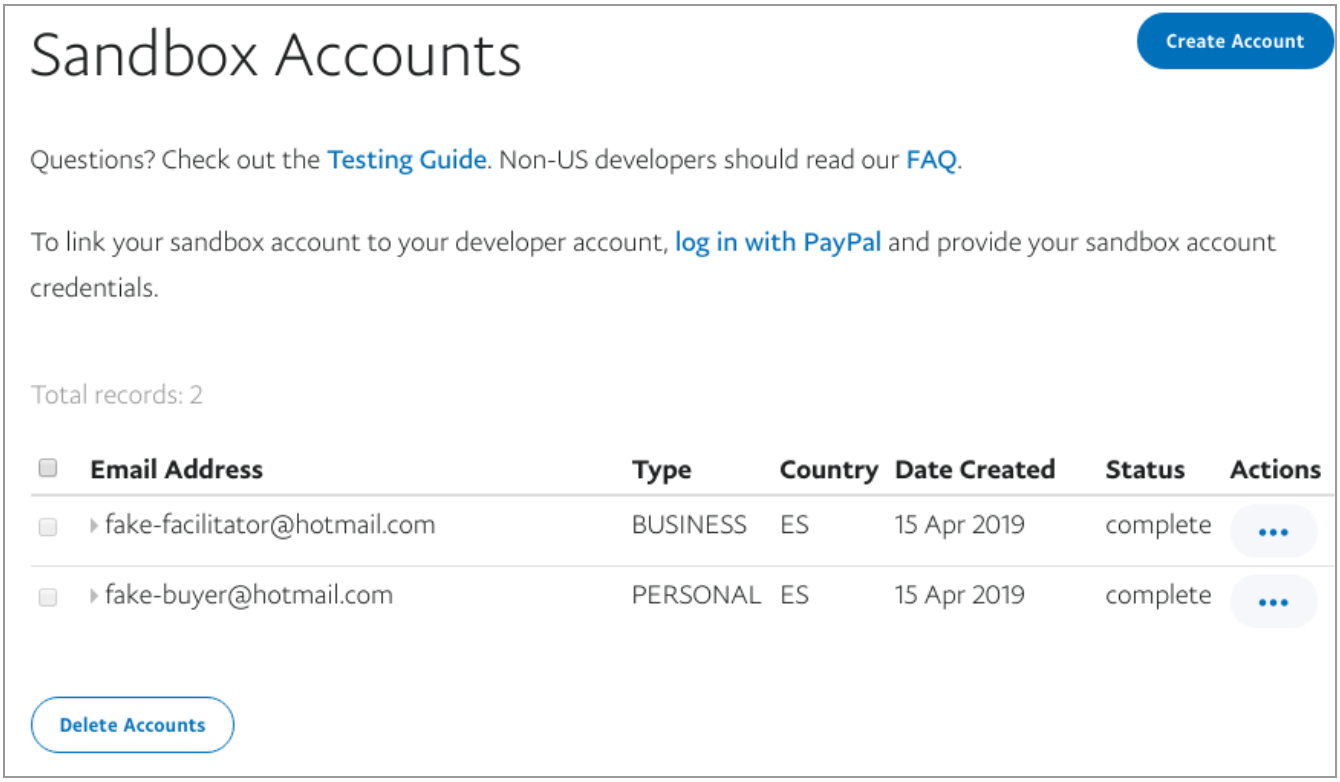


Imagen 42. Sección de creación de cuentas falsas en el panel de desarrollo de Paypal.

Con tal de añadir Paypal a la aplicación, tuve que registrar un nueva cuenta en la web principal y seguidamente ya disponía de acceso al apartado de desarrolladores, donde se pueden hacer varias operaciones como por ejemplo crear cuentas falsas para probar si el proceso de pago funciona, tal y como muestra la imagen 42.

En el centro de desarrolladores se obtiene una clave con la que gracias al consumo de una API en el servidor back-end, podemos realizar una llamada para comprobar si un identificador de compra es correcto o no, además de información diversa como la cantidad de dinero, comprador y demás. La idea es permitir donar al donante directamente con Paypal con un botón en la aplicación web que abra una venta de Paypal y una vez la donación esté hecha, el cliente realice una llamada al servidor back-end

con el identificador de compra para comprobar si la compra es correcta o no. De esta forma, el proceso de compra queda totalmente compacto de cara a los usuarios de la aplicación.

Finalmente, adjunto la imagen 43 donde se puede ver todo el proceso desde que el usuario presiona el botón de pagar, hasta que recibe el mensaje final de la transacción.

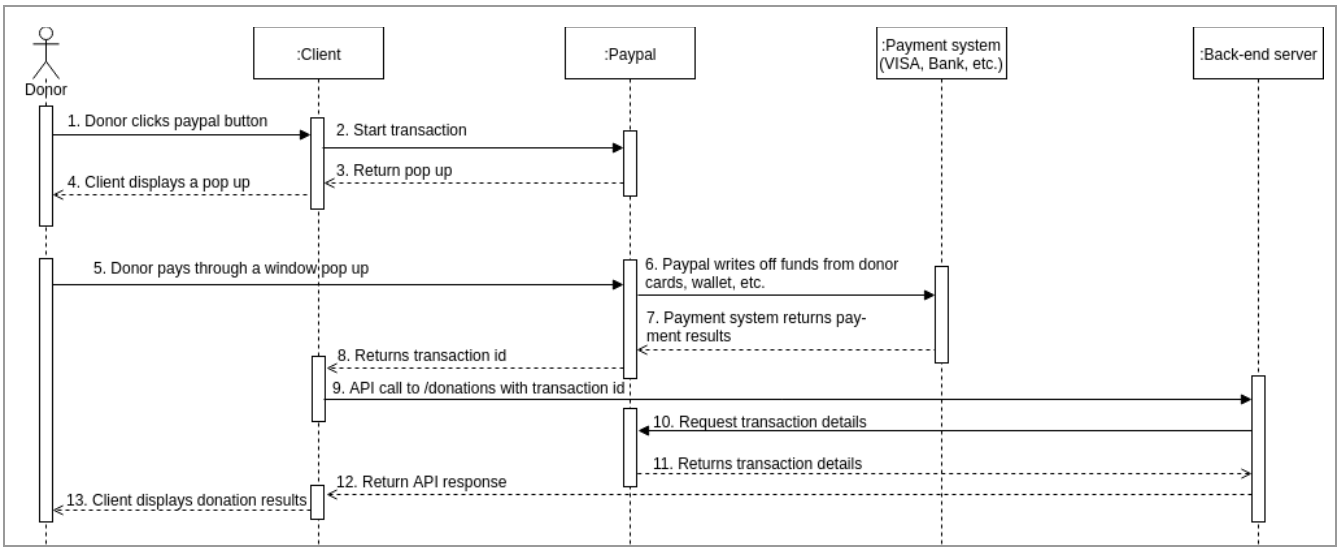


Imagen 43. Diagrama de secuencia - Ejemplo de donación con Paypal.

1. El donante hace click en el botón de Paypal.
2. El cliente realiza una petición HTTP POST a la API de Paypal: */v2/checkout/orders*.
3. Paypal devuelve la información necesaria para crear un pop up.
4. El cliente enseña una ventana pop up al donante.
5. El donante introduce información de de pago en la ventana pop up.
6. Paypal retira dinero del donante, ya sea de la tarjeta u otro tipo de método de pago a través de un sistema de pago.
7. El sistema de pago devuelve el resultado de la transacción a Paypal.
8. Paypal devuelve un identificador de compra al cliente.
9. Se realiza una llamada HTTP POST a la API que hemos especificado en el apartado de controladores: */donations*.
10. El servidor back-end pide detalles de la transacción a través de la API de Paypal: */v2/checkout/orders/{id}*.
11. Paypal devuelve los detalles de la compra en formato JSON.
12. El back-end devuelve el estado de la donación como respuesta a la llamada del paso 9.
13. El cliente muestra un mensaje del estado de la transacción al donante.

Bitly

Bitly es otro de los servicios usados por el servidor back-end. Se trata de un generador de enlaces que lo único que hace es, a partir de un enlace, bitly genera otro con su propio dominio y ruta. El hecho de usar Bitly se debe a querer facilitar el acceso al regalo, a los receptores de regalos que acceden desde una carta. De esta forma tendrán que escribir menos para acceder al regalo, y además, aumentamos la misterio de la carta.

Este servicio se utiliza en el momento en el que se crean los regalos. Si recordamos la imagen 37 del diagrama de clases, tenemos la clase regalo (gift) que tiene un atributo llamado url secreta (secret_url), donde esta url es el enlace generado por Bitly.

Sin embargo, hay que tener en cuenta que estamos usando el plan gratuito, que limita la generación de enlaces a 1000 al mes, por lo que este servicio solo se utiliza en producción. Es decir, en el entorno de desarrollo y pruebas, se utiliza el enlace original al regalo.

En la imagen 44 de abajo, vemos un diagrama que describe la utilización de este servicio. Con tal de incluir una capa más de abstracción y facilitar el uso de la API de Bitly, se ha usado una librería que encapsula las llamadas a HTTP.

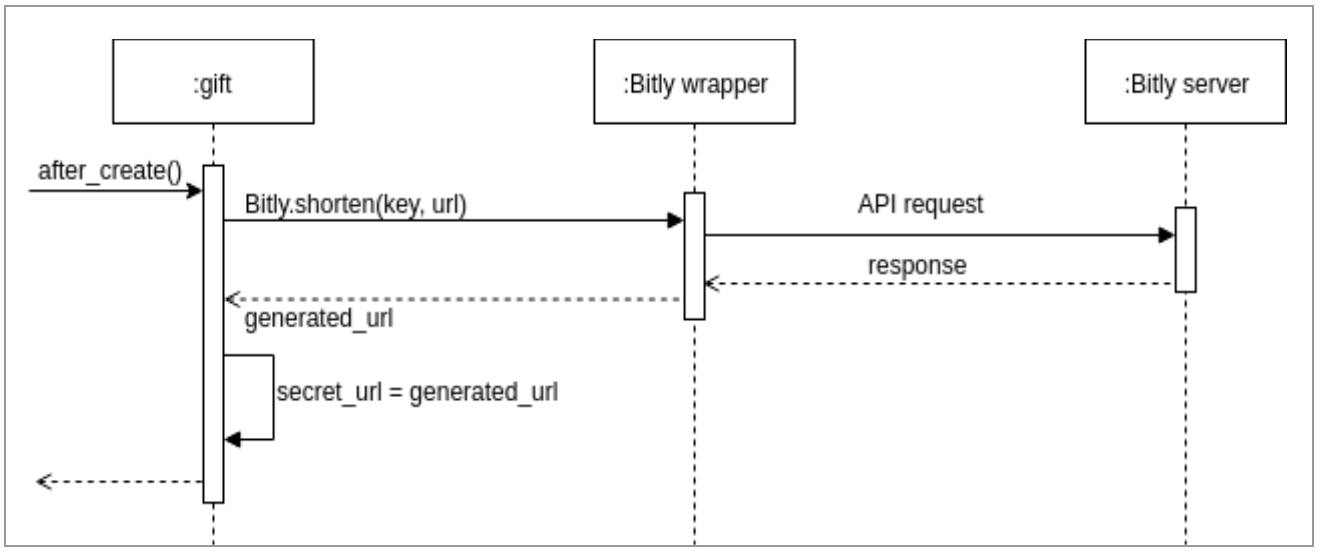


Imagen 44. Diagrama de secuencia - Generación de enlace Bitly.

Como podemos ver, primero tenemos el regalo (Regalo), que ejecuta una función llamada after_create . Ésta función se llama después de que la instancia regalo se ha creado de forma inmediata. Lo primero que hace es llamar a la librería de terceros descrita en el párrafo anterior y, a partir de una llave identificadora proporcionada por Bitly (key) y de una url, la librería hace una llamada HTTP a la API de Bitly para así obtener una respuesta (response) que contiene la url generada. Una vez tenemos esta url generada, solo tenemos que actualizar el atributo secret_url por ésta url.

Mail

La inclusión de un servicio mail es necesario para poder realizar las historias de usuario *Recibo de compra*, *Notificación de regalo abierto* e *Invitación al panel de administración*. En cada una de éstas historias de usuario es necesario enviar un mensaje e-mail, y es por ello que se han buscado servicios que ofrecen una manera de poder enviarlos.

Por suerte, en Ruby on Rails, se pueden crear clases que derivan de una clase base llamada ActionMailer, donde se puede definir los mensaje que se quieren enviar. En las clases generadas para enviar e-mails se especifican los campos típicos de la generación de un e-mail tales como asunto o e-mail al que se quiere enviar y también las variables que se quieren usar dentro del mensaje. Seguidamente se llama a renderizar una vista en un lenguaje llamado ERB (que sintácticamente es HTML con formas de acceder a parámetros), por lo que esta plantilla es renderizada para generar HTML/CSS plano.

Una vez hecha la clase que deriva de ActionMailer con los templates definidos, hacía falta configurar el servidor SMTP por el cual se enviarán los e-mails. Para ello, encontré un servicio llamado SendGrid, que tiene un plan gratuito con el que se pueden enviar 100 e-mail al día. Lo único que tenía que hacer era configurar ActionMailer con los datos del servidor SMTP proporcionado por SendGrid, además de un usuario y contraseña.

Parámetros usados para acceder al servidor SMTP:

- Dirección IP: Dirección de red que hace referencia al servidor proporcionado por SendGrid.
- Puerto: Número de puerto por el que se establece la conexión.
- Usuario: Nombre de usuario proporcionado por SendGrid.
- Contraseña: Contraseña de identificación proporcionado por SendGrid.
- Dominio: Dominio que se conectará al servidor SMTP. Esto se hace para que no cualquier persona pueda conectarse al servidor.

Una vez configurado el acceso al servidor con los parámetros especificados ante, ya podía enviar mensajes, pero era un poco incómodo tener que usar un e-mail real, por lo que pensé en hallar otra forma de ver enviar mensajes cuando me encuentre en el entorno de desarrollo y pruebas, y hablando sobre ésto con un compañero de la oficina, me recomendó un servicio llamado Mailtrap.

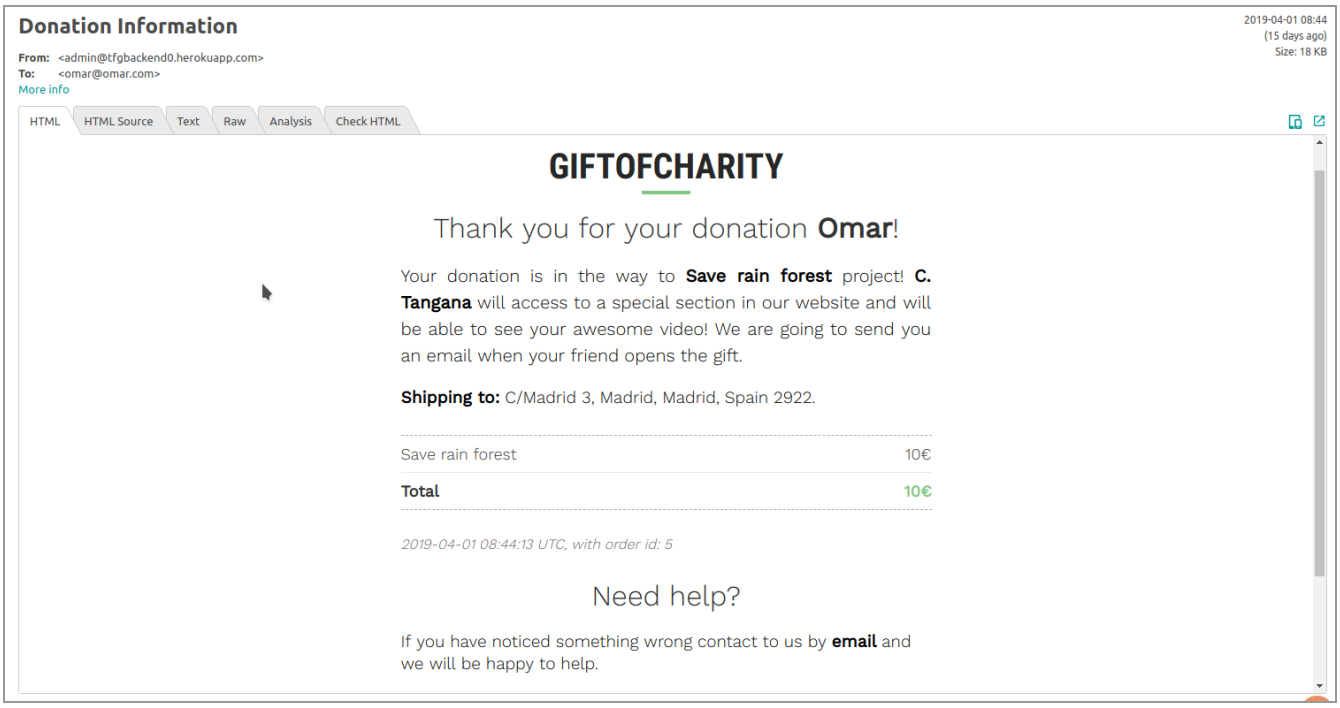


Imagen 45. E-mail capturado por Mailtrap de la historia de usuario: *Recibo de compra*.

Mailtrap es otro servicio que ofrece un servidor SMTP aislado únicamente para pruebas, de manera que de igual forma a cómo hemos configurado ActionMailer con SendGrid, realizamos el mismo proceso de forma simétrica, pero ésta vez para el entorno de desarrollo y pruebas. De ésta forma, cuando se ejecute la aplicación de forma local, podemos utilizar e-mails inventados y éstos irán a parar a la web de mailtrap, donde tenemos un límite de 5000 e-mails al mes con el plan gratuito.

Firebase

Si bien en Heroku tendremos dos proyectos en dos servidores distintos. En estos servidores no se pueden almacenar vídeos o imágenes como si de un servicio de almacenamiento se tratase. Es por ello que tuve que buscar un servicio donde almacenar archivos multimedia de forma que pudiese acceder a ellos a través de algún enlace.

Poco después de haberme planteado esta necesidad en el proyecto, me acordé de un servicio llamado Amazon S3 que permite almacenar datos de forma segura y escalable [26]. Sin embargo, este servicio es de pago, y si bien dispongo de una cuenta de estudiante para acceder a servicios de amazon, no dispongo de permisos suficiente para utilizar Amazon S3 desde una API.

Una vez descartado Amazon, recordé un servicio llamado Firebase mantenido por Google, en el que una de las cosas que ofrece es un servicio de almacenamiento a través de una API. La idea es configurar Firebase desde la web a través de unas reglas y ser llamado directamente desde el cliente. Es decir, no está pensado para ser llamado desde servidores, por lo que podemos llamar a este

servicio directamente desde el cliente en vez de pasar por el servidor back-end, tal y como se ve en la imagen 4 del apartado Infraestructura.

Si hacemos un poco de memoria, hay una historia de usuario llamada *Subir vídeo,* donde se ha de poder subir un vídeo para el regalo del receptor. Además, teniendo en cuenta que en el diagrama de clases de diseño hay un atributo llamado video_url para la clase regalo (gift), la idea es subir el vídeo a Firebase y obtener el enlace al vídeo, de manera que el proceso de donación se realice con éste enlace.

```
service <<name>> {
  // Match the resource path.
  match <<path>> {
    // Allow the request if the following conditions are true.
    allow <<methods>> : if <<condition>>
  }
}
```

Imagen 46. Especificación de reglas en Firebase.

Usar Firebase como medio de almacenaje sin tener que pasar los datos por un servidor intermediario hace que haya que especificar un filtro con tal de que no todos los ficheros se puedan subir. Lo bueno de Firebase es que tiene un apartado donde se pueden especificar reglas de manera sencilla en un lenguaje parecido a JSON. Las reglas pensadas a introducir son las siguientes:
- Vídeos han de ser menores a 200mb de tamaño (Lo cual hace cumplir el requisito no funcional de capacidad).
- Nombre del vídeo menor a 100 caracteres.
- El formato del archivo ha de ser de vídeo.

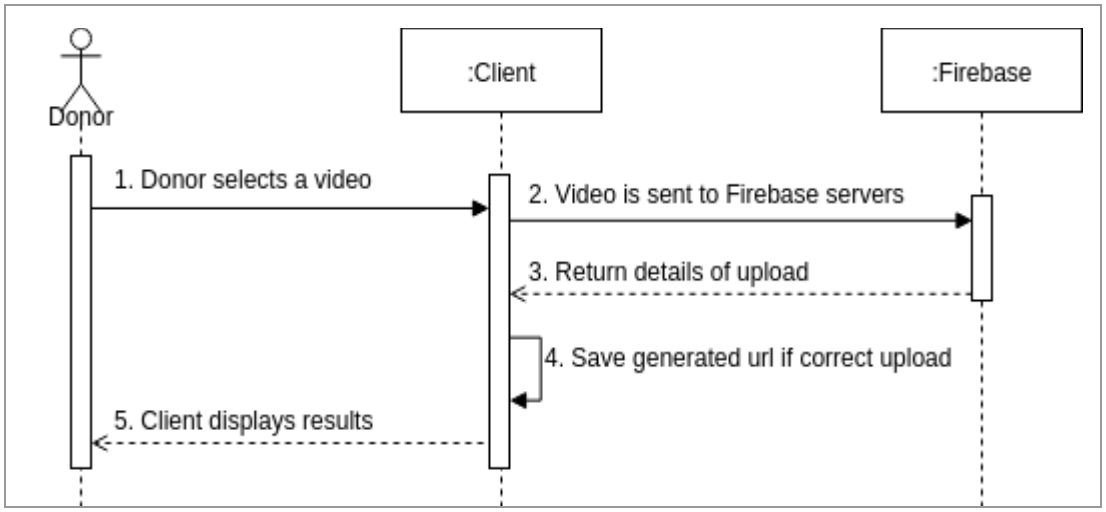


Imagen 47. Diagrama de secuencia - Ejemplo de subida de vídeo a Firebase.

En la imagen 47 podemos ver el proceso de subida de vídeo en forma de diagrama de secuencia. Este diagrama enseña el primer paso del proceso de donación. La descripción paso por paso es la siguiente:
1. El donante selecciona un vídeo que desea que el receptor del regalo vea.
2. El vídeo es enviado a Firebase para que se almacene en sus servidores.
3. El cliente recibe los detalles de la subida del vídeo, que contendrá el enlace del vídeo si el archivo enviado a Firebase es correcto.
4. El cliente almacena la url generada para pasos posteriores en el proceso de donación si los detalles de la subida del vídeo indican que no ha habido error alguno.
5. El cliente muestra una previsualización del vídeo si todo ha ido bien y deja al donante seguir con el proceso de donación. En caso contrario, el donante tiene la posibilidad de subir un nuevo vídeo.

Por último, decir que en este proyecto se ha usado el plan gratuito de Firebase que permite almacenar como máximo 5gb. Si bien hoy en día es un cantidad muy pequeña, para este proyecto, que se acerca más a la definición de prototipo, va perfecto. Sin embargo, lo bueno de Firebase es que permite escalar los servicios que ofrece a través del panel de control que tiene, por lo que si se quisiese aumentar el tamaño de almacenamiento se podría hacer de manera transparente a los usuarios y a los desarrolladores del proyecto, sin embargo se tendría que desembolsar dinero.

8.3.5. Adaptaciones

Hay algunas historias de usuario que pueden ser completadas mediante la adaptación de librerías/proyectos de código abierto. Partiendo de la base de que el tiempo que dispongo no es indefinido, he pensado que es una buena idea ya que aceleramos el cumplimiento de las diversas historias de usuaio definidas en la sección de los requisitos del sistema.

Cuestionarios

Una parte de los regalos a los receptores son los cuestionarios, por lo que en esta subsección nos centraremos en la historia de usuario *Consumir cuestionario*.

Con tal de no reinventar la rueda y añadir complejidad al proyecto, en una reunión con el director de la empresa, éste me comentó sobre otro proyecto en el que utilizaban un servicio para generar cuestionarios. Éste servicio llamado Typeform[29] me permitía crear cuestionarios desde un panel y obtener un enlace a éstos. Tiene una interfaz muy sencilla para crear cuestionarios y es bastante personalizable. Sin embargo, para utilizar todas las funcionalidades que me interesaban con tal de adaptar este servicio al proyecto necesitaba un plan mejor que el básico, que brindaba pocas funcionalidades.

Si bien la empresa me ofreció invertir un poco en este servicio, decidí buscar otras opciones y encontré un proyecto de código abierto en Github. Éste proyecto llamado ‘quick-quiz’ se trataba de una web

[29] https://www.typeform.com/

hecha con HTML, CSS, JS, es decir, tecnología relativamente sencilla, que a través de un archivo de configuración, presentaba una cuestionario con diseño minimalista [27].

Adaptación

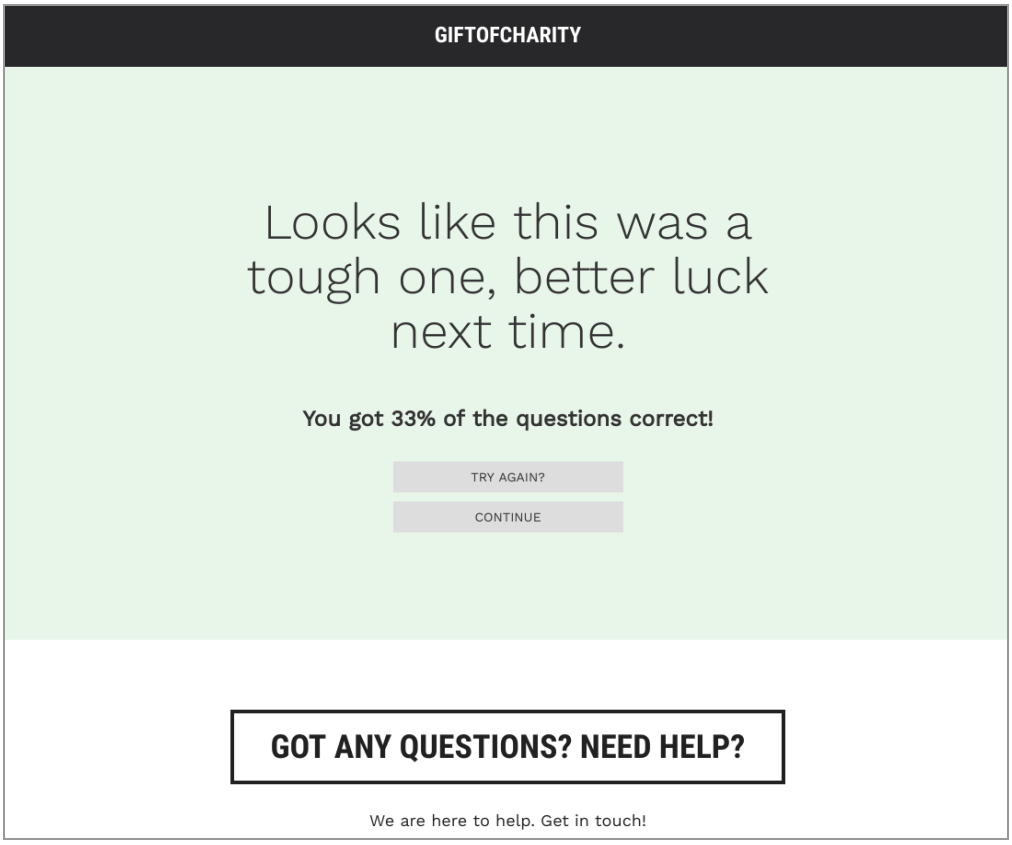


Imagen 48. Pantalla final del cuestionario.

La idea era adaptar este proyecto a la aplicación web en React que estaba realizado. Lo bueno de 'quick-quiz' es que a través de un archivo de configuración JSON se cargan todas las preguntas y respuestas del cuestionario. Por lo que si conseguía que el back-end devolviese un JSON con un correcto formato, tan solo tendría que hacer que quick-quiz cargue el cuestionario devuelto.

Después de pensar diversas soluciones, decidí empezar con la adaptación de quick-quiz. Que podemos resumir en diversos pasos:

- Primero tuve que configurar el back-end para que a partir de una ruta, devolviese un cuestionario con las propiedades exactas que interpreta el quick-quiz. Entonces, monté la base de datos y me encargué de realizar la funcionalidad de la ruta de la API que a partir de un identificador devuelva el cuestionario correctamente formateado para su interpretación.

- Seguidamente, me encargué de que el proyecto quick-quiz estuviese integrado en la aplicación web hecha con React, de manera que tuviese su ruta. Lo que hice fue coger el proyecto quick-quiz y ponerlo en una carpeta donde se suelen subir ficheros estáticos en React (public). De manera que

cuando los receptores entrasen a su regalo, acceden al cuestionario a partir de un elemento HTML llamado ‘iframe’, en el que es otra ventana independiente dentro de una misma pestaña del navegador. De esta manera se pueden tener dos tecnologías diferentes en un mismo proyecto interactuando de forma independiente.

- Lo siguiente que hice fue pensar en la comunicación entre las ventanas dichas en el paso anterior. Me documenté sobre la forma de pasar mensajes de ventana padre a ventana hijo, donde la ventana principal sería la aplicación en React y la ventana hijo el iframe y, a través de mensajes, la ventana padre manda el cuestionario en formato texto (string) y el iframe lo interpreta y serializa a objeto JS con tal de que el cuestionario cargue correctamente. Ésto último lo tuve que modificar del proyecto quick-quiz, ya que obviamente había que adaptarlo.

- Una vez tenía los cuestionarios funcionando, quedaba tocar el diseño de los cuestionarios. Debido a mi experiencia en proyectos de éste tipo con tecnologías sencillas, pude cambiar el ‘look and feel’ de los cuestionarios a través de CSS, de manera que concuerde con el estilo de la aplicación web en React.

- Finalmente, coloqué un botón extra donde se muestra la pantalla de resultados del cuestionario, de manera que al hacer click en él, se mande un evento a la ventana padre y se cargue la otra parte del regalo. En la imagen 48 vemos el final de un cuestionario, con el botón antes mencionado.

Funcionamiento

Una vez explicada la adaptación, en la imagen 49 podemos ver una imágen con dos escenarios. El primero muestra el proceso de carga de un cuestionario y el segundo, el mensaje enviado por la ventana iframe una vez es cuestionario está finalizado.

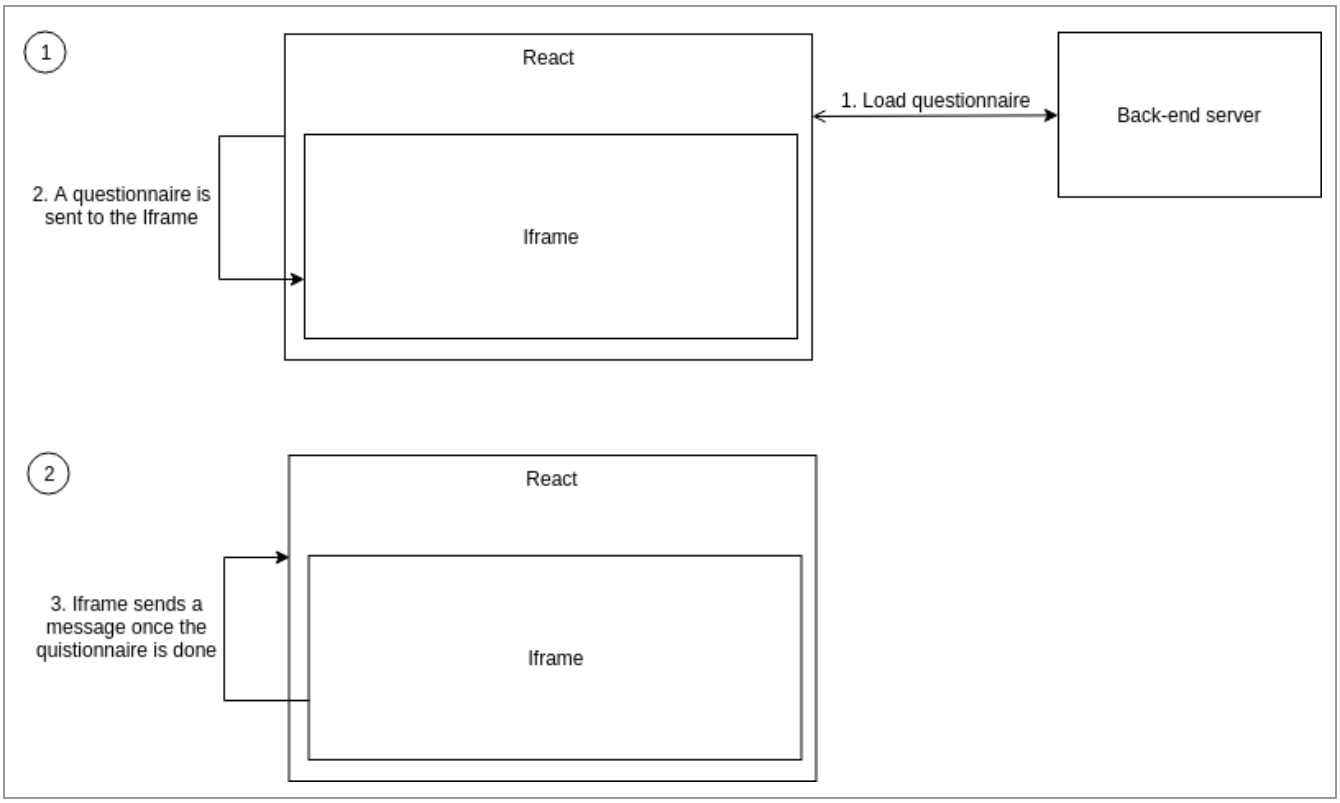


Imagen 49. Funcionamiento de la adaptación de 'quick-quiz' al proyecto.

1. Cargamos el cuestionario haciendo una llamada a la API que ofrece el servidor back-end con un identificador para obtener el cuestionario correcto.
2. Una vez tenemos el cuestionario, enviamos un mensaje con el cuestionario en formato de texto a la venta hijo, es decir, al iframe.
3. Cuando el receptor del regalo acaba el cuestionario y pulsa el botón para finalizar, el iframe manda un mensaje en forma de aviso a la ventana padre, es decir, la ventana principal que contiene la aplicación web.

Panel de administración

Tal y como se muestra en la imagen 4 de la sección de infraestructura, existe una relación entre los administradores y el servidor back-end. Lo que indica esta relación es el acceso al panel de administración y en esta subsección voy a explicar cómo se ha implementado.

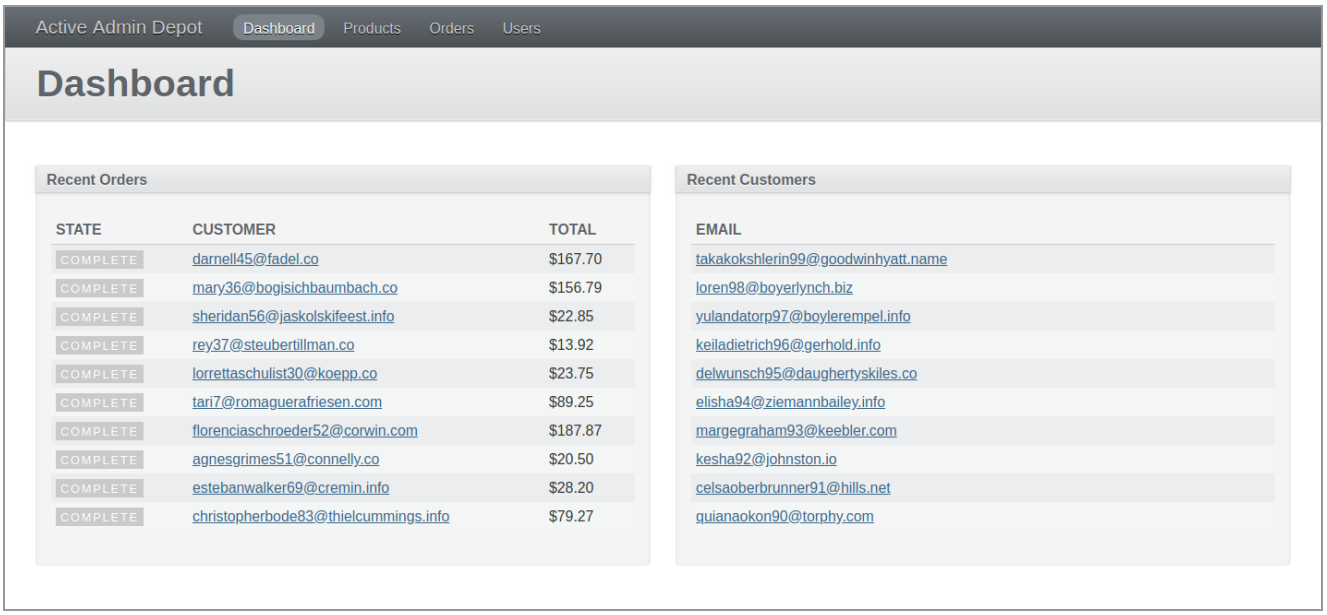


Imagen 50. Demo de Active Admin.

Active Admin

Active Admin es un complemento para Ruby on Rails de código abierto. El panel de administración generado incluye varias funcionalidades como [28]:

- Barra de navegación: Un elemento que te permite atravesar las diversas vistas del panel. Siempre disponible en la parte superior del panel de administración.

- Autentificación: Genera un inicio de sesión, cerrar sesión, recuperar contraseña. Para usar esta funcionalidad tuve que instalar otra librería llamada: Devise [29], muy famosa para facilitar la tarea de autenticación en aplicaciones web. Con esta librería generé un modelo administrador, que es el que se usa para acceder al panel de administrador.

- Índices: Se puede generar un índice para cada modelo del sistema. Este índice suele tener una tabla o lista con todos las instancias de la base de datos con información básica y paginados.

- API y descarga: Debajo del índice se sitúan unos botones en los que se puede descargar y acceder a la información en diversos formatos como JSON, CSV o XML. (El proyecto prescinde de esta funcionalidad).

- Filtros: En el lado derecho de los índices se muestra un panel donde se puede filtrar instancias a través de filtros según los atributos de los modelos.

- Acciones: Se pueden añadir botones, enlaces u otros contenidos en la parte superior de las vistas. Estas acciones pueden hacer referencia a los elementos de los índices o a cada uno de ellos por separado.

- Operaciones CRUD: Por defecto, este panel de administración permite realizar operaciones como crear, borrar, actualizar y leer en los modelos.

Personalización

El hecho de usar Active Admin hace que generar un panel de administración sea sencillo y rápido, pero por otra parte, personalizaciones pequeñas han requerido más tiempo de lo necesario, y es por eso que hace falta un proceso de aprendizaje con tal de personalizarlo y dejarlo listo para producción.

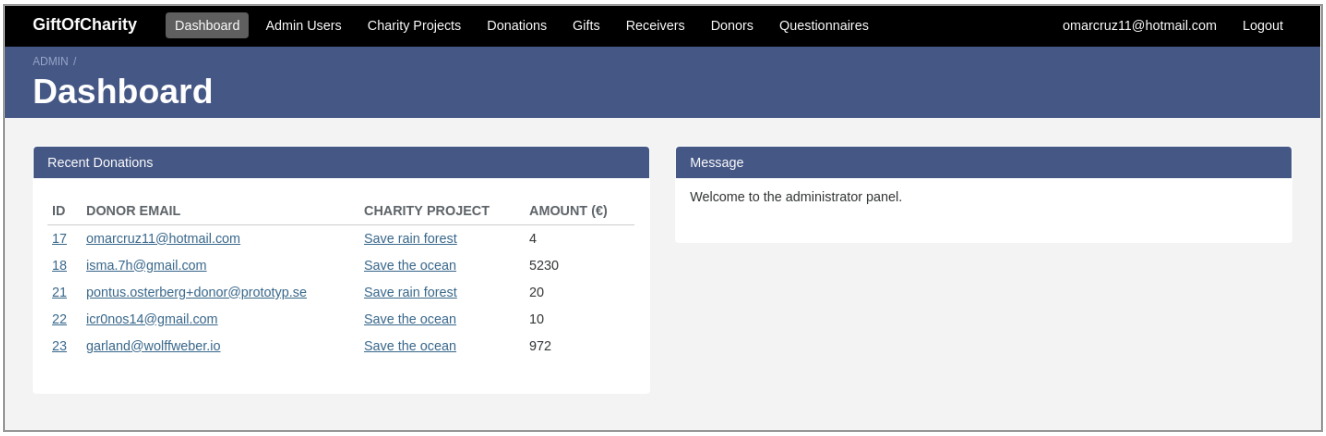


Imagen 51. Vista principal del panel de administración.

Lo primero que hice al añadir Active Admin al proyecto fue cambiar los estilos con tal de que mejorase de forma visual. Como podemos ver, en la imagen 50 destacan los colores grises y apagados, mientras que en la imagen 50, vemos un panel de administrador más moderno y con colores más acordes. Este cambio ha sido posible gracias a otra librería llamada ActiveAdminTheme [30], que a través de una inyección de estilos CSS, permite cambiar varios de los elementos del panel de administración. Sin embargo, aún habían cosas que no me gustaban, y tuve que modificarlas manualmente con css.

Otra de las cosas que se cambiaron fue en el índice de los proyectos de caridad. Éste índice contenía para cada instancias, los valores de los atributos que hay almacenados en la base de datos. Si bien para los otros modelos de la base de datos ésto era suficiente. En el caso de los proyectos de caridad se veía el enlace de la imagen que tiene asociada de forma literal. Por lo que decidí añadir una columna al índice y añadir un elemento imagen que representa la imagen que tiene asociado cada proyecto de caridad. En la imagen 25 de la sección de la descripción de pantallas se puede ver este cambio.

Pasando un poco a la estructura del panel de administración. He comentado antes que se genera un índice para cada modelo. Éste índice vendría siendo como la vista principal, sin embargo, hay modelos en la base de datos que no hace falta que tengan esta vista exclusivamente. El modelo que define las preguntas de los cuestionarios (question) y las respuestas (answer) no se han introducido en el panel de administración para llevar un control exclusivo, ya que no se ha visto interés alguno ni estaba descrito en las historias de usuario.

Active Admin facilita las operaciones de crear y actualizar mediante un formulario para todos los modelos. Inicialmente, estos formularios están disponibles para todos los modelos, sin embargo se han tenido que restringir en algunos, pues no eran necesarios. Por ejemplo tenemos el modelo regalo (gift), que no tiene sentido poder crearlo sin una donación relacionada (aunque luego sí puedas relacionarlo). Además, crear un donante y recibidor por separado para luego relacionarlo con una donación resultaba poco práctico, por lo que decidí personalizar el cuestionario de crear donaciones. De manera que se pueda rellenar un formulario con toda la información que facilita el proceso de donación que se haya en el front-end. Y ésto último se puede ver en la imagen 28 de la sección de descripción de pantallas.

Siguiendo con los cambios en los formularios, otro formulario que se tuvo que cambiar fue el de los cuestionarios. Como comenté antes, los modelos pregunta y respuesta no tienen un índice ni una vista para éstos. Y la falta de estos índices hace necesario crear un formulario más avanzado en el apartado de cuestionario. De manera que se pudiesen crear preguntas y respuestas anidadas, tal y como se enseña en la imagen 35 de la descripción de pantallas.

Finalmente, respecto al modelo administrador, se ha modificado la forma en que éste funciona. Originalmente, el panel de administración permite crear un adminsitrador a partir de un e-mail y una contraseña. Sin embargo, si tenemos en cuenta la historia de usuario *Invitación al panel de administración*, es el nuevo administrador quien debe poner su propia contraseña. Es por eso que se ha quitado la contraseña del formulario de creación del nuevo administrador y se ha dejado sólamente la inserción de un e-mail. De esta forma, se ha introducido un cambio de manera que a través del e-mail proporcionado se genera un mensaje donde el nuevo administrador, a partir de un enlace único, puede rellenar un formulario donde puede introducir su propia contraseña para acceder al sistema. Para la vista del enlace, se ha aprovechado la funcionalidad de cambiar contraseña.

Funcionalidades extra

Si bien Active Admin proporciona un kit de funcionalidades para cada modelo. Hay algunas historias de usuario que no están cubiertas. Es por ello que se ha tenido que añadir algunas funcionalidades a este panel de administrador.

Id	Sent	Seen	Secret Url	Receiver	Donation	Opened At	Created At	
30	YES	YES	http://bit.ly/2WKkKCl	Aurelio Lind	23	June 13, 2019 09:00	June 12, 2019 12:16	View Edit
29	NO	NO	http://bit.ly/2ZjCK34	Anderson Hand Sr.	22		June 12, 2019 12:02	View Edit

Imagen 52. Enlaces con funcionalidades extra en el lado derecho de los elementos de la lista de regalos.

Si recordamos el capítulo de especificación de las historias de usuario, tenemos el *marcar regalo como enviado (adm)*. Esta historia de usuario está 'medio resuelta' con Active Panel, ya que podemos editar para cada instancia del modelo regalo el atributo enviado (sent). Sin embargo, también resulta poco

intuitivo, por lo que decidí añadir un enlace con el que se pueda marcar una instancia de regalo en cada una de los elementos aparecen en el índice del modelo regalo.

Por otro lado, tenemos la historia de usuario *Descargar carta (adm)*. Para añadir esta funcionalidad, primero creé un enlace que tienen cada una de las instancias en el índice del modelo regalo (gift), y a partir de este enlace, puse la lógica para que, a partir de un template, se generase un pdf con la ayuda de la librería PDFkit [31]. De esta forma, los administradores pueden descargar la carta en formato PDF de cada uno de los regalos del sistema.

8.4. Capa de gestión de datos

Esta última capa tendrá una base de datos objeto - relacional (ORDBMS). La idea ha sido usar un framework MVC que proporcione una herramienta muy de moda llamada: ORM. Un ORM es una biblioteca especializada en acceso a datos que genera una abstracción para acceder a los datos como si de clases se trataran [32]. Es decir, no tendría que lidiar con el lenguaje de bajo nivel que se utiliza en las bases de datos.

Por lo que en esta sección, voy a explicar de forma sintetizada cómo se ha especificado la base de datos con el ORM de Ruby on Rails llamado Active Record y además, la base de datos objeto-relacional usada.

8.4.1. Active Record

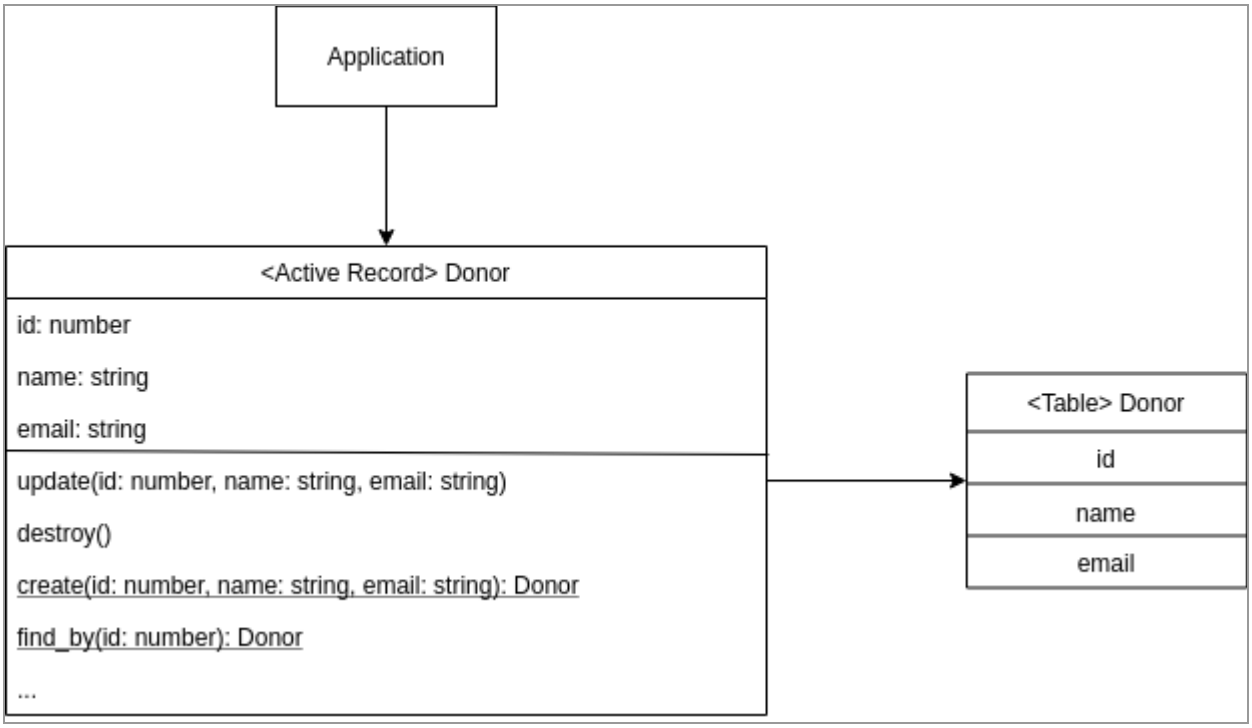


Imagen 53: Active Record pattern con el modelo donante (Donor).

Active Record de Ruby on Rails es una implementación del patrón Active Record, que de hecho, es la descripción de un sistema con mapeo objeto-relacional (ORM) [33]. En el patrón Active Record, los objetos llevan datos persistentes y además tienen funciones que permiten operar sobre éstos, como los objetos en la programación orientada a objetos tradicional. De esta forma, podemos operar con los objetos de forma directa, y éstos se actualizarán en la base de datos sin la necesidad de escribir sentencias (querys) a bajo nivel.

Para usar Active Record en Ruby on Rails, tuve que familiarizarme con el concepto de 'migraciones'. Las migraciones permiten que la base de datos pueda crecer en el tiempo. En vez de definir todo el esquema de la base de datos en un solo archivo, se hacen modificaciones de las tablas en archivos llamados migraciones que están identificados por un timestamp. En estas migraciones se describen todos los cambios a las tablas en una clase que hereda de la clase *ActiveRecord::Migration* y tiene una función donde se han de especificar los cambios que se quieren hacer [34].

Aparte de las donaciones, están los modelos, que son las clases donde se pueden definir atributos calculados, validaciones en los atributos, relaciones con otros modelos, funciones auxiliares, disparadores y demás. Estas clases se crean para cada tabla de la base de datos. Por ejemplo, en la imagen 53 vemos el modelo donante (donor) en una clase que tiene diferentes métodos establecidos por Ruby on Rails como *update, destroy*, *create* o *find_by.*

Finalmente, decir que si bien usar el ORM de Ruby on Rails requiere un proceso de aprendizaje de los diferentes conceptos que hay, el hecho de no usarlo habría hecho que el proyecto se hubiese alargado más de la cuenta, además de tener que poner mucha más énfasis en la base de datos.

8.4.2. Bases de datos objeto-relacional usadas

En el momento en el que se empieza a usar Ruby on Rails, se utiliza una base de datos objeto-relacional llamada SQLite. Esta base de datos es muy simple, ya que prácticamente consta de un fichero dónde se guarda todo. Debido a su sencillez y configuración por defecto como base de datos, la he estado usando para toda la etapa de desarrollo y pruebas.

A pesar de que SQLite viene configurado y resulta ser sencillo de usar, no se puede usar para producción, pues no tiene características robustas como la concurrencia ni es escalable. Es por eso que tuve que buscar otro tipo de base de datos objeto-relacional para el entorno de producción.

Si recordamos el apartado de infraestructura, comenté que se ha utilizado un complemento de Heroku llamado Heroku Postgres. En la web de Heroku, en el apartado del proyecto que corresponde al servidor back-end, seleccioné el complemento antes mencionado con tal de obtener unas credenciales de acceso al servidor que contiene la base de datos PostgreSQL, de esta manera, tan solo tuve que introducir las credenciales obtenidas en la configuración del proyecto de Ruby on Rails, en el apartado de configuración del entorno que va a producción.

De no haber utilizado el complemento de Heroku Postgres, podría haber montado otro servidor en el que estuviese cualquier base de datos objeto-relacional que conociese. De hecho, el ORM de Ruby on Rails permite configurarlo con varios tipos de bases de datos como MariaDB, Oracle Database o incluso bases de datos no relacionales como MongoDB. Sin embargo, hubiese vuelto a escoger PostgreSQL, pues es el tipo de base de datos relacional que aprendí en la asignatura BD y con el que más familiarizado estoy.

9. Implementación

En este capítulo nos centrará en las tecnologías usadas para desarrollar los proyectos que están en los servidores back-end y front-end, además de las herramientas utilizadas.

9.1. Servidor front-end

9.1.1. Frameworks valorados

Antiguamente, los programadores trabajaban con HTML (HyperText Markup Language) para la estructura de la web, CSS (Cascading Style Sheets) para el diseño y JS (Javascript) para hacer la web dinámica mediante scripts. Sin embargo, todo lo que se escribía era interpretado directamente por el navegador web, lo que que dependiendo del programador, el resultado no era eficiente o no era compatible con todos los navegadores. Además, con el paso de los años se utilizan cada vez más librerías, y con este modo de programar, tienes que adjuntar un enlace hacia la librería en el HTML para utilizarlo con los scripts en JS, lo cual se hacía muy pesado y mezclaba lenguajes.

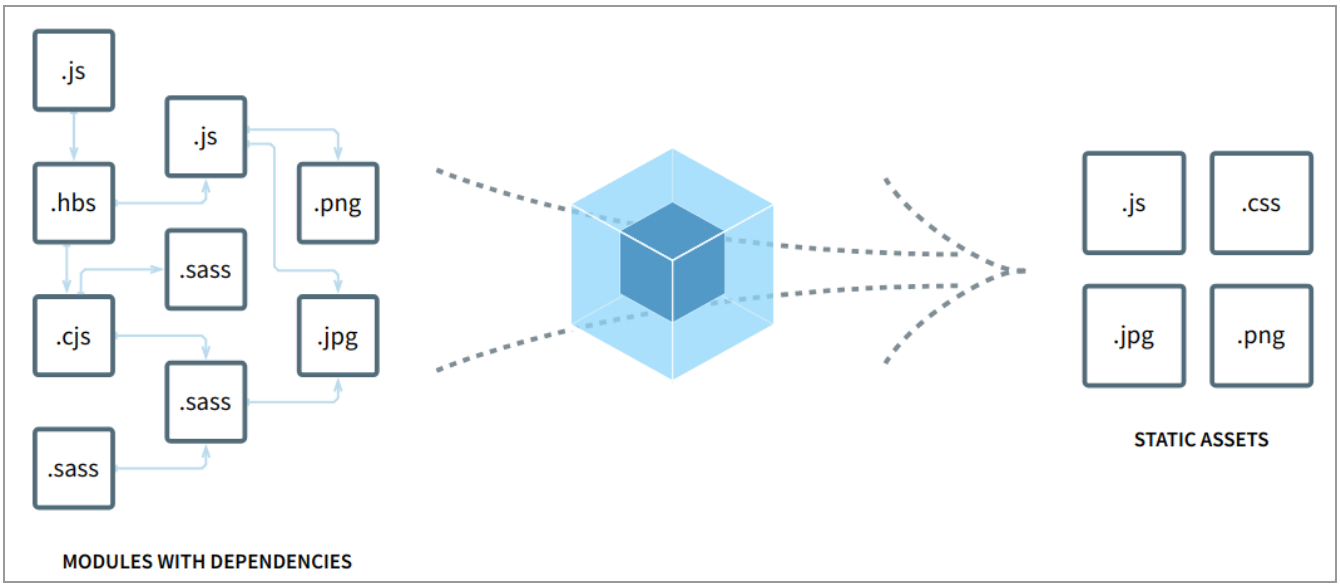


Imagen 54. Funcionamiento del empaquetador de módulos Webpack.

Hoy en día, se suelen utilizar transpiladores que permiten pasar de un código en un lenguaje JS moderno o incluso otro lenguaje de programación hacia un JS compatible con todos los navegadores. Por si eso no fuera poco, los transpiladores suelen estar configurados dentro de empaquetadores de módulos que a partir de diversas dependencias y complementos escritos en otros lenguajes de programación o de marcado, permiten generar solo archivos HTML, CSS y JS, de hecho, podemos ver en la imagen 54, un ejemplo de un empaquetador de módulos llamado Webpack, que es el más utilizado actualmente.

Habiendo introducido un poco esto, para el servidor front-end se ha buscado una librería o framework que me permitiese realizar un proyecto escalable y claro. Y gracias a las tecnologías dichas antes, hay un gran abanico de posibilidades, sin embargo, voy a centrarme en las tres más populares.

Angular[30]

Angular es un framework desarrollado por Google que permite crear aplicaciones SPA (Single Page Application). Las principales finalidades de Angular son: Separar el lado cliente del servidor, de tal manera que puedan desarrollarse en paralelo; separar la lógica de la manipulación del DOM; introducir pruebas en el lado del cliente [35].

Si bien Angular es un framework bastante completo y robusto. Desde mi punto de vista, está centrado para grandes aplicaciones que tienen que ser escalables, donde hay muchas personas desarrollando simultáneamente. Además, la curva de aprendizaje suele ser un poco elevada.

Vue[31]

Vue es un framework de código abierto que al igual que Angular, permite crear aplicaciones SPA. Es famoso por tener una curva de aprendizaje corta, tener una buena documentación y flexibilidad para complementar el framework con diversas librerías. Sin embargo, Vue tiene una comunidad de desarrollo cerrada, no es tan famoso como Angular. Además, muchos de los usuarios que usan Vue no hablan inglés, por lo que hay menos oportunidades de encontrar problemas resueltos en internet [36].

React[32]

Finalmente tenemos React, se trata de una librería desarrollada por Facebook. Sirve para crear componentes y renderizarlos en el DOM. Junto a React, se suelen usar un conjunto de librerías que incrementan el potencial que éste tiene. React tiene una gran comunidad detrás, además de ofrecer mucha flexibilidad y tener una curva de aprendizaje media [37].

Como punto negativo, es un poco verboso y muchas personas consideran negativa la flexibilidad que ofrece. Además, las convenciones de uso de React aún se están estableciendo, no hay una metodología estándar definida.

Conclusión

Habiendo investigado las diversas opciones que tenía, hablé con algunos miembros de la oficina y me recomendaron React debido a su curva de aprendizaje media. Entonces, teniendo en cuenta esto y que además podría consultar con los compañeros de la oficina, decidí escoger React para el proyecto.

[30] https://angular.io/
[31] https://vuejs.org/
[32] https://reactjs.org/

9.1.2. Material UI

Para agilizar el proceso de desarrollo de la aplicación web. Hice uso de una librería que ofrece componentes compatibles con React. Estos componentes están hechos para que puedan usarse directamente en el código, de manera que gracias a esta librería, he podido evitar el proceso de diseño de la aplicación desde 0. Sin embargo, los componentes son estándar y predominan los colores azules, por lo que tuve que modificar estos componentes para adaptarlos a mi gusto.

Con tal de personalizar los componentes antes dichos, hice uso de los conocimientos que tenía de proyectos anteriores de CSS, por lo que no fue demasiado complicado.

Componentes

```
<Button onClick={this.toggleDialog(this.constants.ERROR)} color="primary">
    Ok
</Button>
```

Imagen 55. Componente 'Button' de Material UI.

Material UI ofrece componentes varios componentes compatibles con React. Estos componentes funcionan de forma aislada y, al igual que todos los componentes de React, se pueden especificar propiedades (props) con tal de personalizarlos como por ejemplo vemos en la imagen 55, donde tenemos el componente 'Button' al que se le especifican dos propiedades: 'onClick' que toma como valor una función que se ejecuta cuando se hace click en el componente, mientras que 'color' tiene como valor un string que indica los colores que tendrá el botón. El valor de 'color' no es arbitrario, en la web de Material UI está especificado qué valores exactos puede tomar.

Partiendo del ejemplo del párrafo anterior, he de comentar que en el desarrollo del proyecto hice mucho uso de la documentación que ofrece la web de Material UI, pues ahí es donde se encuentran ejemplos de uso y se explica en detalle las props de todos los componentes.

Plantilla

Imagen 56. Diseño inicial de GiftOfCharity.

En las primeras iteraciones del proyecto me centré en el back-end y usé componentes de Material UI para ir haciendo el proyecto desde 0. No me basé en ninguna otra web, sino que añadía los componentes mínimos necesarios e iba avanzando el proyecto. Sin embargo, una vez ya tenía el back-end bastante avanzado y la aplicación ya tenía lógica, tuve un reunión con el director de la empresa y me comentó que la interfaz no era muy amigable y que debería hacerla más moderna. Debido a esa reunión decidí centrarme en el diseño durante una iteración completa y obtuve una web mucho más profesional e intuitiva, cosa que hizo que los requisitos no funcionales de percepción se vieran cumplidos.

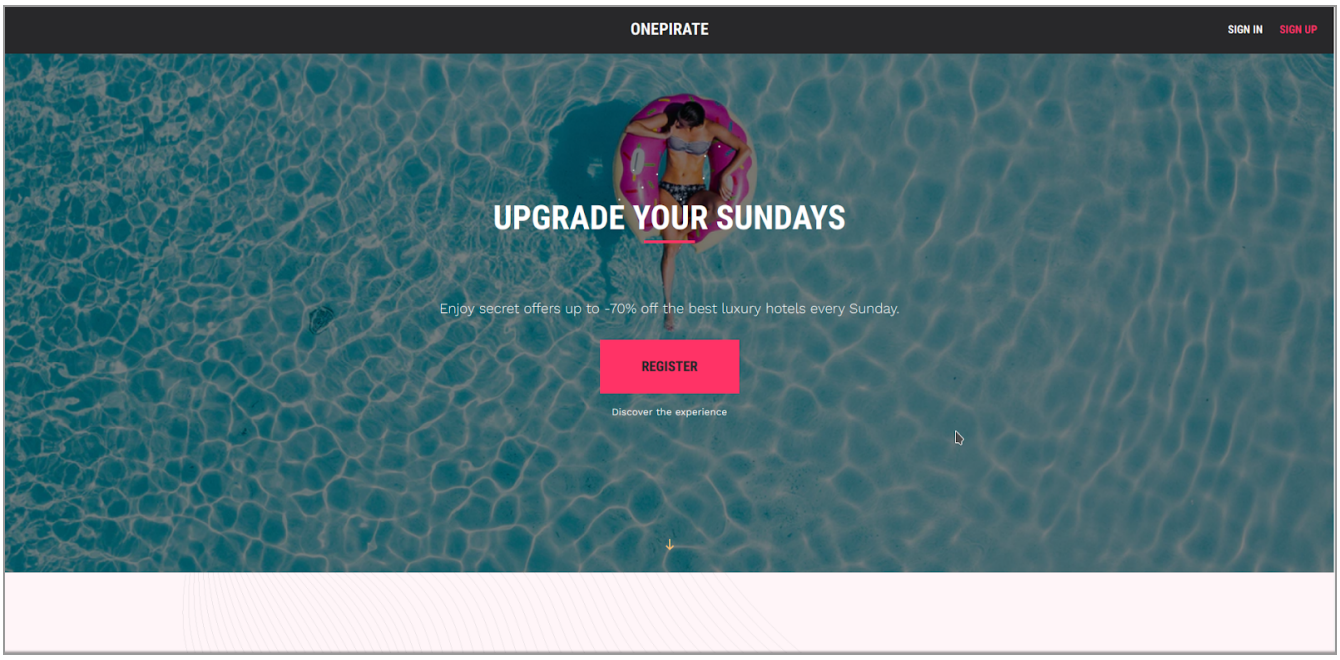


Imagen 57. Captura de la plantilla ‘Onepirate theme’.

Para conseguir que la aplicación fuese más moderna, busque plantillas que usasen React y encontré que en la misma web de Material UI había un apartado llamado ‘premium themes’ donde se encontraba una plantilla que contenía una sola página principal con botones, cabecera, pie de página y alguno que otro componente. No me lo pensé dos veces y decidí usar esta plantilla llamada ‘Onepirate theme’ para la aplicación web, que si bien no disponía de muchos componentes, los que habían estaban muy bien y me sirvió para cambiar la manera de diseñar los componentes futuros, como por ejemplo el componente llamado ‘stepper’ de la imagen 58, donde muestro la conversión de un componente plano de Material UI a uno propio que concuerda con el ‘look and feel’ de la aplicación.

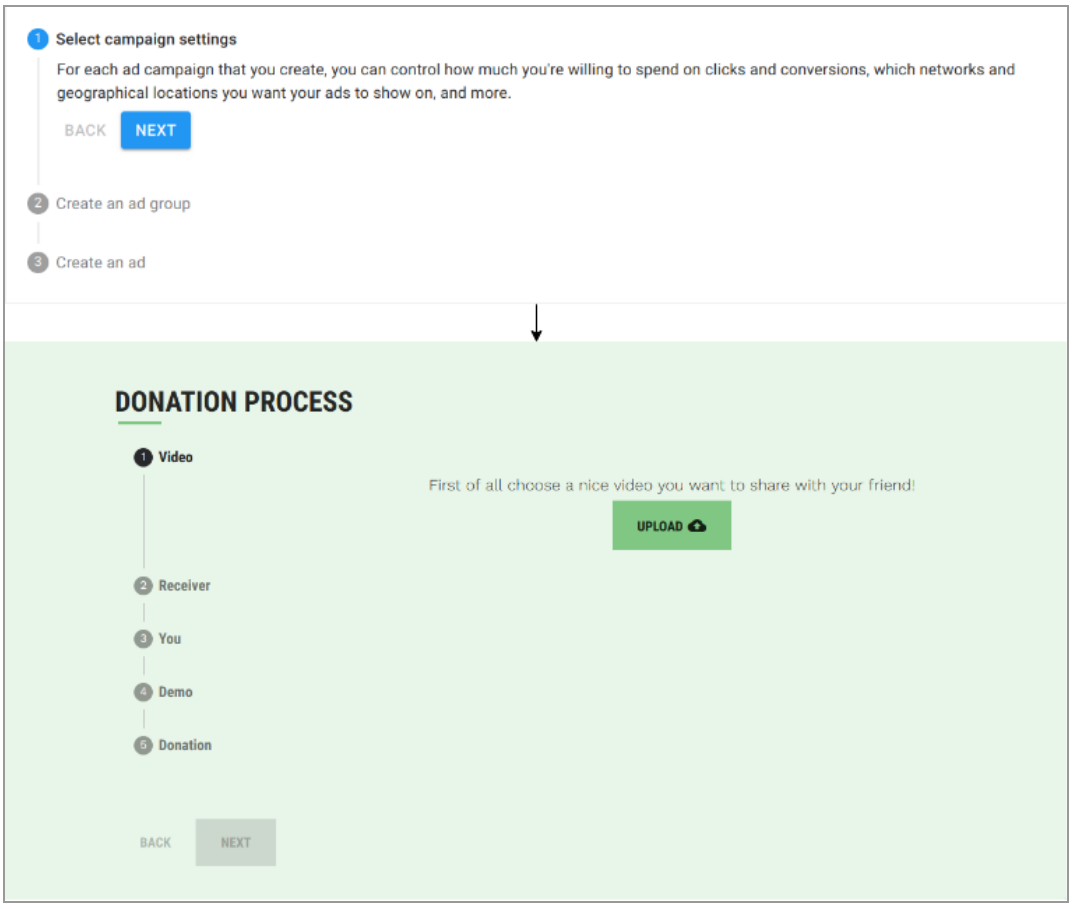


Imagen 58. Componente 'stepper' antes y después. Arriba el elemento por defecto y abajo después de la personalización.

9.1.3. Diseño web adaptativo

Si recordamos la imagen 4 del capítulo de diseño, los donantes y receptores tienen como cliente un ordenador y un móvil. El hecho de que haya puesto un móvil hace referencia a este apartado, donde explico la compatibilidad de la aplicación web con los móviles. No desde un punto de vista de técnico, sino la compatibilidad en lo que a la interfaz se refiere.

Imagen 59. Diseño adaptativo para diferentes dispositivos [38].

Hoy en día muchas de las aplicaciones web son desarrolladas aplicando una técnica llamada 'responsive design' o diseño adaptativo. Esta técnica se basa en construir la aplicación web de manera que sea adaptable a dispositivos de diferentes tamaños como tablets y smartphones [39]. El hecho de realizar un diseño adaptativo en esta aplicación hace que el público que lo utilice se expanda, pues se puede acceder desde más dispositivos ofreciendo la misma experiencia. Además, reduce la necesidad de realizar otra aplicación nativa en paralelo para éstos otros dispositivos, lo que disminuye costes de desarrollo y mantenimiento.

Componente Grid

Con de realizar un diseño adaptativo, hice uso de un componente llamado 'Grid' de la librería de Material UI. Este componente permite encapsular otros componentes dentro y, dependiendo del valor de las propiedades que tiene, cambia su tamaño.

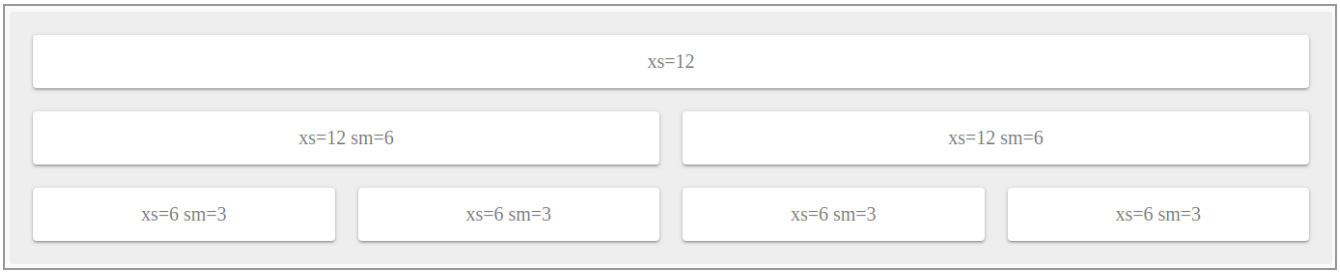


Imagen 60. Ejemplo de diseño con el componente Grid.

```
<Grid container spacing={24}>
  <Grid item xs={12}>
    <Paper className={classes.paper}>xs=12</Paper>
  </Grid>
  <Grid item xs={12} sm={6}>
    <Paper className={classes.paper}>xs=12 sm=6</Paper>
  </Grid>
  <Grid item xs={12} sm={6}>
    <Paper className={classes.paper}>xs=12 sm=6</Paper>
  </Grid>
  <Grid item xs={6} sm={3}>
    <Paper className={classes.paper}>xs=6 sm=3</Paper>
  </Grid>
  <Grid item xs={6} sm={3}>
    <Paper className={classes.paper}>xs=6 sm=3</Paper>
  </Grid>
  <Grid item xs={6} sm={3}>
    <Paper className={classes.paper}>xs=6 sm=3</Paper>
  </Grid>
  <Grid item xs={6} sm={3}>
    <Paper className={classes.paper}>xs=6 sm=3</Paper>
  </Grid>
</Grid>
```

Imagen 61. Código correspondiente al ejemplo de la imagen 30.

Los elementos Grid se dividen dos tipos: contenedores (containers) e ítems, donde los ítems han de estar dentro de un elemento padre contenedor, que define la anchura máxima. Como vemos en la imagen 61 que es el código correspondiente a la imagen 60, los grid tipo ítems tienen una propiedad llamada 'xs' y 'sm', éstas propiedades pueden tomar valores del 1 al 12, donde 12 representa expandirse a todo lo que el contenedor deje, mientras que 1 representa el expandirse una doceava parte de lo que miden los otros ítems adyacentes. Partiendo de esto, decir que los valores de xs y sm no se usan simultáneamente, sino que dependiendo del tamaño de la ventana del dispositivo que use la aplicación, se usará uno u otro. Xs y sm son llamados breakpoints y, aunque el ejemplo muestra estos dos, hay otros como md, lg y xl. Cada uno se utiliza dependiendo del tamaño de la ventana en píxeles de la manera siguiente [40]:

- xs, extra-small: 0px o más grande
- sm, small: 600px o más grande
- md, medium: 960px o más grande
- lg, large: 1280px o más grande

- xl, extra-large: 1920px o más grande

Tal y como hemos visto, Grid es un componente sencillo de utilizar y permite generar diseños responsivos de forma rápida, sin centrarse en reglas condicionales de CSS, que es como años atrás se solía hacer.

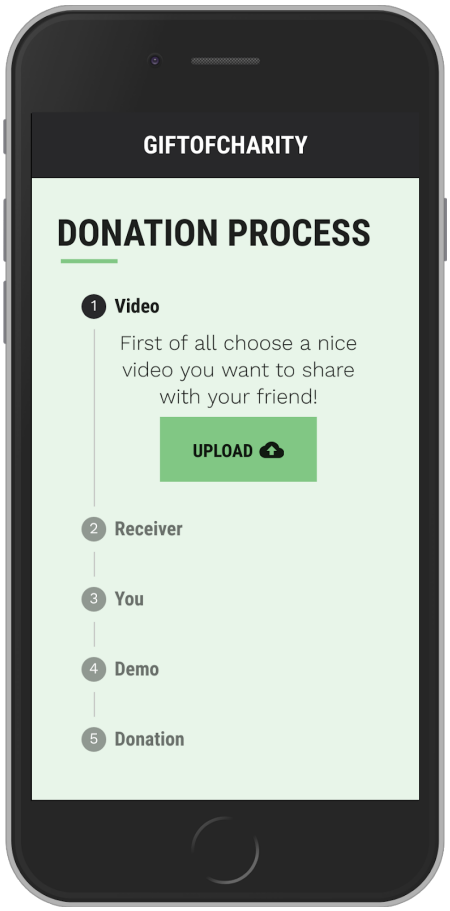


Imagen 62. Proceso de donación desde un dispositivo móvil Iphone 6.

Finalmente, cabe mencionar que si bien se ha utilizado la técnica de diseño adaptativo, el mínimo tamaño de pantalla para el que se ofrece la mejor experiencia es para dispositivos con 375 x 667 píxeles, ya que si bien Grid facilita mucho el trabajo, hay detalles que he tenido que hacer manualmente y el hecho de hacerlo compatibles con absolutamente todos los dispositivos del mercado hubiese hecho que el proyecto se hubiera alargado más de la cuenta.

9.1.4. Botón Paypal

En esta sección voy a explicar cómo se conecta el cliente a Paypal. Si recordamos el apartado Paypal de la sección de servicios del capítulo de diseño, hay un diagrama que está en la imagen 43, donde se puede ver cómo el cliente hace click en un botón en el cliente, y éste acaba abriendo un pop up desde donde el donante puede pagar (pasos 1 - 5). Para generar éste botón, la documentación de Paypal

ofrece una forma mediante la inyección de un script en JS. Sin embargo, puesto que el cliente se ha desarrollado con React, hice uso de una librería llamada 'react-paypal-express-checkout' de tal manera que abstrae las llamadas a Paypal y permite configurar un componente para insertarlo directamente en la aplicación web.

```
<PaypalBtn
  client={this.state.client}
  currency="EUR"
  total={parseInt(this.state.fields.amount.value)}
  onSuccess={this.onSuccess}
  onError={this.onError}
  style={{
    shape: 'rect',
    size: 'medium',
    fundingicons: true,
  }}
/>
```

Imagen 63. Componente de la librería 'react-paypal-express-checkout'.

Como podemos observar en la imagen 63, la inserción del botón Paypal se hace mediante un componente al que se le añaden distintas propiedades:
- client: Se trata de un objeto JS que define los identificadores que hacen referencia a la cuenta de Paypal.
- currency: La moneda del pago.
- total: Cantidad de dinero que mostrará el pop up y que el usuario pagará.
- onSuccess: Función que se llama si el pago a través del pop up es correcto.
- onError: Función que se llama si el pago a través del pop up no es correcto.
- style: Estilos definidos por la documentación de Paypal [41] para personalizar el botón (no nos olvidemos que ésta librería es una capa de abstracción, por lo que internamente, hace uso del script en JS que proporciona Paypal antes mencionado).

Como último detalle, mencionar que la función ejecutada en la propiedad 'onSuccess', es la que se encarga de llamar a la API del servidor back-end con la ruta /donations, tal y como muestra el paso 8 del diagrama de la imagen 43.

9.1.5. Reglas Firebase

```
1  service firebase.storage {
2   match /b/{bucket}/o {
3     match /videos {
4       // Cascade read to any image type at any path
5       match /{allVideos=**} {
6         allow read;
7       }
8
9       // Allow write files to the path "videos/*", subject to the constraints:
10      // 1) File is less than 200MB
11      // 2) File name (stored in imageId wildcard variable) is less than 100 characters
12      // 3) Content type is a video (mp4, etc)
13      match /{videoId} {
14        allow write: if request.resource.size < 200 * 1024 * 1024
15                     && videoId.size() < 100
16                     && request.resource.contentType.matches('video/.*')
17      }
18    }
19  }
20 }
```

Imagen 64. Especificación de reglas para la funcionalidad *Subir vídeo*.

Si recordamos el capítulo de diseño, se usa un servicio de Firebase en la capa de dominio con tal de que el usuario donante pueda subir un vídeo para el receptor y éste se almacene en un servidor. Sin embargo hacían falta unas reglas con tal de que hubiera una especie de filtro y no se permitiese al usuario donante subir lo que quieras al servidor.

En la imagen 64 vemos que se ha hecho uso de la funcionalidad para especificar reglas que otorga Firebase. De forma resumida, podemos decir que en la línea 14 se define que los vídeos sean menores a 200MB, en la línea 15 que los nombres de los vídeos sean menores a 100 caracteres y finalmente en la línea 16 sólo se permiten archivos formato vídeo.

9.2. Servidor back-end

9.2.1. Frameworks valorados

Para este servidor se ha buscado un framework que siguiese un patrón MVC. De esta manera, se usaría la parte del modelo para la base de datos, los controladores que serían la API y las vistas se generarían sólo para el panel de administrador, pues recordemos las tres capas del capítulo de diseño, este servidor no se centrará en la parte visual, sino que se centrará en la lógica de la aplicación.

Hoy en día hay cientos de frameworks MVC que elegir, pero entre todos ellos he valorado principalmente a tres.

Sails[33]

[33] https://sailsjs.com/

Sails es el primer framework que utilicé para realizar una aplicación web y el primero a tener en cuenta. Utiliza node.js, y posee varias capas de abstracción para que su uso sea sencillo. Dispone de un ORM llamado Waterline y te permite crear una API REST de forma muy sencilla.

Si bien es un framework que conozco bien, su popularidad es un poco baja y no dispone de una herramienta potente para crear paneles de administrador. Además, si bien es cierto que avanzaría más rápido, perdería la oportunidad de aprender a fondo otra tecnología.

Django[34]

Django es otro framework MVC. Es de código abierto y está escrito en Python, un lenguaje muy de moda actualmente. Django fué creado para no tener que lidiar con las mismas operaciones que se enfrentan los programadores al iniciar un proyecto como el inicio de sesión, panel de administración y demás. Es por ello que este framework ofrece herramientas que te permiten usar componentes que ya están listos. [42]

Lo bueno de Django es su panel de administrador. Te genera un login para los administradores, y a partir de los modelos que tienes en el sistema, te permite obtener un panel de administrador eficaz y personalizable.

Ruby on Rails[35]

Ruby on Rails es el framework que utilicé en la asignatura ASW de la carrera. Hice una pequeña aplicación en la parte final del curso y además, es uno de los frameworks principales usados en la empresa en la que desarrollé el proyecto. Es un framework MVC escrito en Ruby, y tiene una gran comunidad de desarrollo detrás. Se han creado muchos sitios web famosos como Twitter o Github y existe una librería compatible llamada Active Admin que sirve para generar paneles de administrador a partir de los modelos que tienes en el sistema, al igual que en Django. Si bien este panel de administrador estéticamente no es muy moderno, tiene una buena documentación donde se puede aprender a personalizar y añadir funcionalidades extra.

Conclusión

Finalmente, me decanté por Ruby on Rails por diversas razones. El punto principal se debe a que es el framework que conocen los integrantes de la empresa donde hago el TFG y el hecho de poder hablar con alguien cuando hay problemas en el desarrollo del software para que te aconsejen, es de gran ayuda. También me ha convencido la librería Active Admin, que se ocupa de algunas de las cosas que tengo definidas en las historias de usuario, pero además, genera funcionalidades adicionales que dan valor a mi aplicación. Por último, decir que también he tenido en cuenta el hecho de tener un nivel mínimo de conocimientos, pues sabía que agilizaría la etapa de desarrollo, pero además, profundizará en esta tecnología.

[34] https://www.djangoproject.com/
[35] https://rubyonrails.org/

9.2.2. Renderizado de vistas

Ruby on Rails es un framework modelo-vista-controlador y como tal, tiene la capacidad de poder generar vistas. En este proyecto, en un principio se decidió prescindir de las vistas de Ruby on Rails debido a que el servidor front-end sería en el encargado de la parte visual de la aplicación. Sin embargo, hay algunas historias de usuario, como "*Recibo de compra*", en las que resulta útil poder renderizar vistas directamente desde el servidor.

```
<tr>
  <td align="justify"
      class="padding-copy r-font"
      style="...">
    Your donation is in the way to <b><%= @donation.charity_project.name %></b> project!
    <b><%=  @donation.get_receiver.name %></b> will access to a special section in our website
    and will be able to see your awesome video! We are going to send you an email when your friend
    opens the gift.
  </td>
</tr>
<tr>
  <td align="left" style="..." class="padding-copy r-font"><b>Shipping to:</b> <%= @donation.get_receiver.address %>,
    <%= @donation.get_receiver.city %>, <%= @donation.get_receiver.province %>, <%= @donation.get_receiver.country %>
    <%= @donation.get_receiver.postcode %>.
  </td>
</tr>
```

Imagen 65. Porción de código escrito en erb para especificar el diseño del e-mail para los recibos de compra.

Para renderizar vistas en Ruby on Rails se utiliza un lenguaje de plantilla llamado 'ERB', donde la idea básica es es que se tiene un archivo html especial en el que se pueden utilizar unas variables definiadas por la función que llama a renderizar la vista. Así, en el archivo antes comentado se pueden usar parámetros y así generar plantillas que se renderizan para generar HTML/CSS plano. En la imagen 65 podemos ver la plantilla en ERB para generar el e-mail de la historia de usuario: *Recibo de compra*.

Además de para los e-mails, se ha utilizado el renderizado de vistas en la generación de pdfs. Si recordamos la historia de usuario "*Descargar carta (adm)*", se ha de generar un archivo PDF que es la carta que reciben los receptores. La idea es básicamente la misma, tener una plantilla donde se sustituyan parámetros y se genere un HTML. Sin embargo, en este caso además de generar el HTML, se ha utilizado una librería para pasarlo a formato PDF.

9.2.3. Esquema de la base de datos

Partiendo del diagrama de clases de diseño, a continuación enseñaré el diseño de las tablas y atributos de la base de datos.

Sin embargo, antes que nada me gustaría comentar los atributos comunes que tienen todas las tablas generadas por el ORM de Ruby on Rails, ya que son campos que se van repitiendo para todas las tablas y no están añadido en la imagen 66.

- created_at: fecha en la que el registro fue creado.
- updated_at: última fecha en la que el registro fue editado.

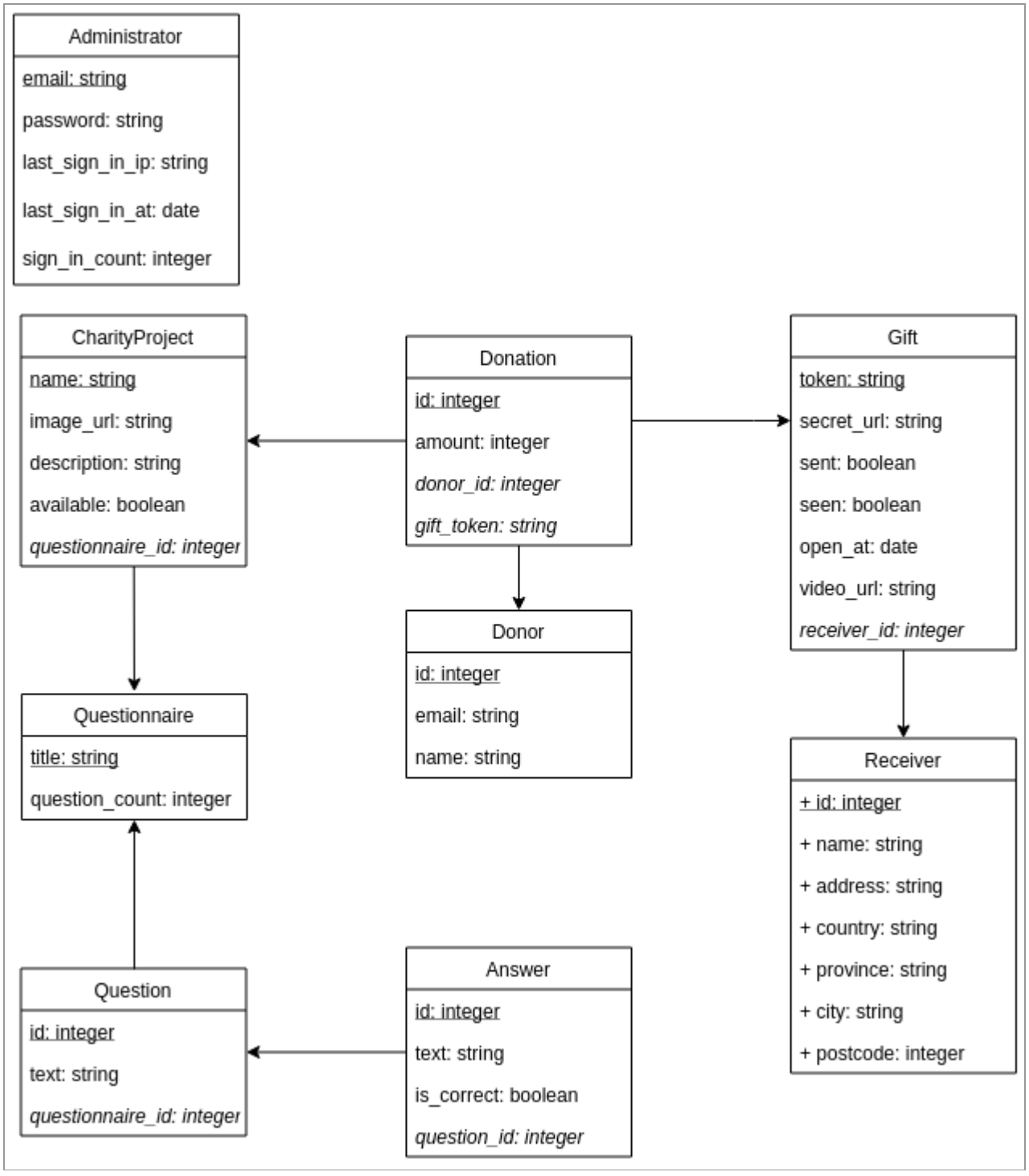


Imagen 66. Esquema de la base de datos.

En la imagen 66 podemos ver el esquema de la base de datos. Los atributos subrayados indican la llave primaria, mientras que los atributos en cursiva representan las llaves foráneas.

El resultado de este esquema parte del diagrama de clases de diseño, sin embargo, en la implementación han aparecido los siguientes nuevos atributos:

Tabla 'Administrator':

- last_sign_in_at: contiene la fecha y hora del último inicio de sesión del administrador.

- last_sign_in_ip: contiene la dirección IP del último inicio de sesión del administrador.

- sign_in_count: indica las veces que el administrador ha iniciado sesión en el sistema.

Tabla 'Questionnaire':

- question_count: indica la cantidad de preguntas asociadas al cuestionario.

9.3. Herramientas

En esta sección se situarán las herramientas utilizadas para llevar a cabo el desarrollo de la aplicación.

9.3.1. IDE's utilizados

Para el desarrollo del proyecto se han usado dos entornos de desarrollo integrados (IDE's) que son productos de la compañía Jetbrains[36]. Éstos productos son de pago, pero se ha hecho uso de una cuenta de estudiante, por lo he podido desarrollar la aplicación sin ningún problema y de forma totalmente gratuita.

[36] https://www.jetbrains.com/

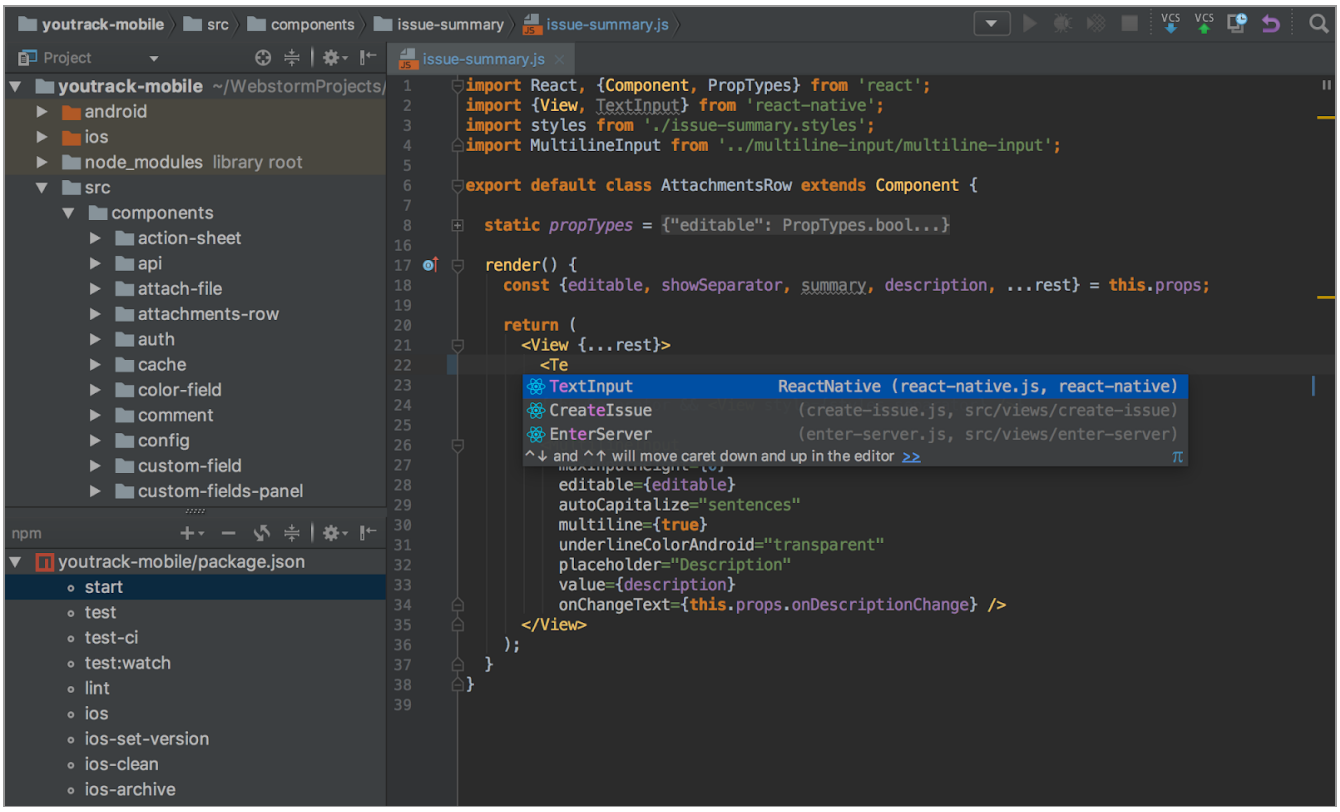


Imagen 67. WebStorm IDE versión 2019.1.1.

- WebStorm: Entorno de desarrollo integrado enfocado en el desarrollo de aplicaciones en JavaScript. Provee de una tecnología ligera para ofrecer soluciones para el desarrollo complejo tanto del lado del cliente como el del servidor [43]. Contiene herramientas de integración, ejecución de tests, completación de código, navegación de clases y demás tecnologías que facilitan mucho el desarrollo.

- RubyMine: Estéticamente igual a WebStorm y muy parecido en cuanto a funcionalidades, pero enfocado a desarrolladores de Ruby y concretamente de Ruby on Rails. Posee también de potentes herramientas que incrementan la productividad durante el desarrollo de aplicaciones. El autocompletado e integración de librerías han sido las funcionalidades que más he sacado proyecto de RubyMine.

9.3.2. Postman

Postman es un software útil para probar peticiones HTTP. Me ha servido para verificar las salidas de las llamadas de la API construida, de tal forma que mediante una sola vista puedo ver todos los detalles referentes a las peticiones hechas.

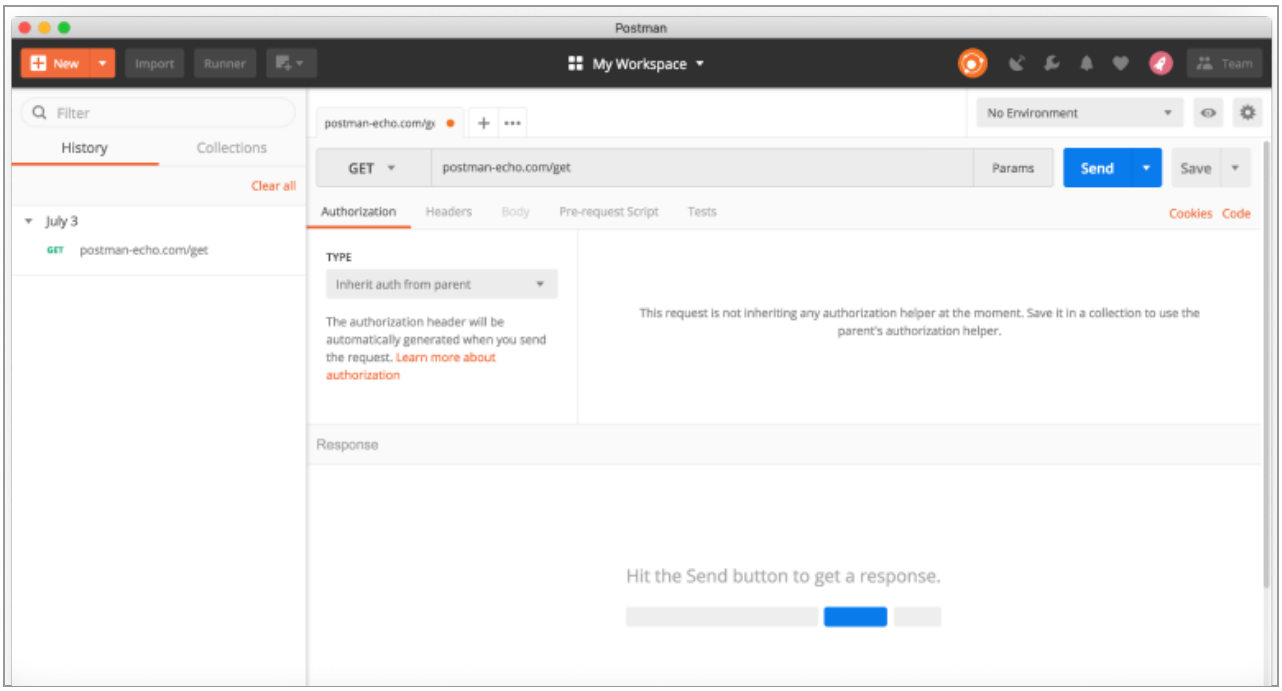
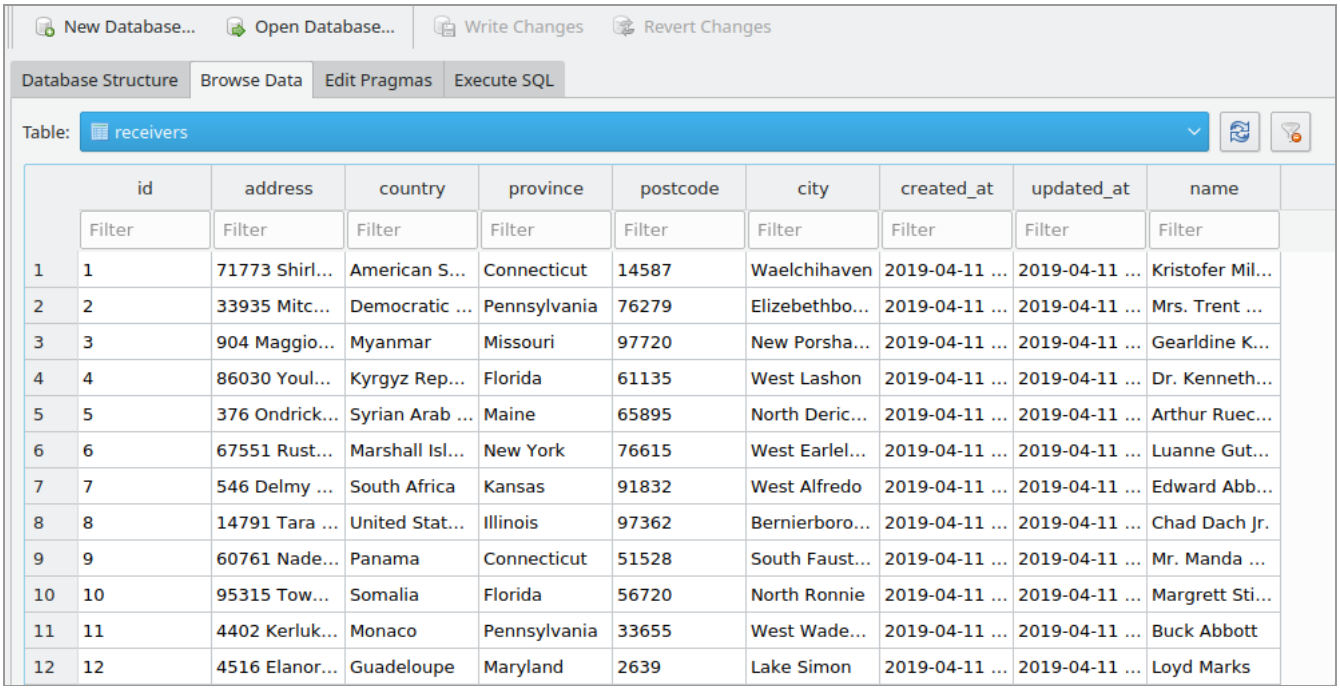


Imagen 68. Postman versión 7.0.9.

9.3.3. SQLiteBrowser



	id	address	country	province	postcode	city	created_at	updated_at	name
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	71773 Shirl...	American S...	Connecticut	14587	Waelchihaven	2019-04-11 ...	2019-04-11 ...	Kristofer Mil...
2	2	33935 Mitc...	Democratic ...	Pennsylvania	76279	Elizebethbo...	2019-04-11 ...	2019-04-11 ...	Mrs. Trent ...
3	3	904 Maggio...	Myanmar	Missouri	97720	New Porsha...	2019-04-11 ...	2019-04-11 ...	Gearldine K...
4	4	86030 Youl...	Kyrgyz Rep...	Florida	61135	West Lashon	2019-04-11 ...	2019-04-11 ...	Dr. Kenneth...
5	5	376 Ondrick...	Syrian Arab ...	Maine	65895	North Deric...	2019-04-11 ...	2019-04-11 ...	Arthur Ruec...
6	6	67551 Rust...	Marshall Isl...	New York	76615	West Earlel...	2019-04-11 ...	2019-04-11 ...	Luanne Gut...
7	7	546 Delmy ...	South Africa	Kansas	91832	West Alfredo	2019-04-11 ...	2019-04-11 ...	Edward Abb...
8	8	14791 Tara ...	United Stat...	Illinois	97362	Bernierboro...	2019-04-11 ...	2019-04-11 ...	Chad Dach Jr.
9	9	60761 Nade...	Panama	Connecticut	51528	South Faust...	2019-04-11 ...	2019-04-11 ...	Mr. Manda ...
10	10	95315 Tow...	Somalia	Florida	56720	North Ronnie	2019-04-11 ...	2019-04-11 ...	Margrett Sti...
11	11	4402 Kerluk...	Monaco	Pennsylvania	33655	West Wade...	2019-04-11 ...	2019-04-11 ...	Buck Abbott
12	12	4516 Elanor...	Guadeloupe	Maryland	2639	Lake Simon	2019-04-11 ...	2019-04-11 ...	Loyd Marks

Imagen 69. Tabla receptor (receiver) en SQLite Browser versión 3.11.2.

En las principales etapas del proyecto cuando no disponía de un panel de administrador, he estado haciendo uso de un software complementario a SQLite que sirve para explorar la base de datos

seleccionando el fichero que lo contiene. El software utilizado para explorar la base de datos se llama SQLite Browser y como vemos en la imagen 69, podemos acceder a una tabla para ver todas las filas que contiene ésta, con sus atributos al descubierto.

10. Aplicación de la metodología

En esta sección de la memoria explicaré la metodología del proyecto aplicada. Para comenzar, he de decir que no ha cambiado mucho respecto a lo planeado en el capítulo de metodología de la memoria, sin embargo, la metodología de trabajo se volvió más completa a medida que aprendía sobre tecnologías y prácticas de los proyectos que hay en la empresa en la que realizo el TFG.

10.1. Integración contínua

Durante las primeras semanas en mi estancia en la empresa oí hablar de herramientas de integración contínua. Entonces recordé la asignatura de gestión de proyectos (GPS) y recuperé mis apuntes para ver si era factible aplicarlo en este proyecto.

La integración continua consiste en realizar integraciones, que son la compilación y ejecución de pruebas de todo un proyecto, de forma automática en un proyecto lo más a menudo posible, de manera que se puedan detectar fallos lo antes posible. [44]

Teniendo en mente la definición y aplicación pensé que era una buena idea aplicarlo tanto el proyecto del servidor back-end como en el del front-end. Con la integración contínua podría:

- Automatizar los tests en un entorno de producción, de manera que el código nuevo subido a Github sea 'seguro'.
- Me aseguraría de que la generación del proyecto del lado de front end se ejecutase correctamente con cada subida de código nuevo a Github, ya que si recordamos el capítulo de implementación, el servidor front-end utiliza un empaquetador de módulos (imagen 54), por lo que con integración contínua nos aseguramos de que el proceso de generación del proyecto se realice correctamente.

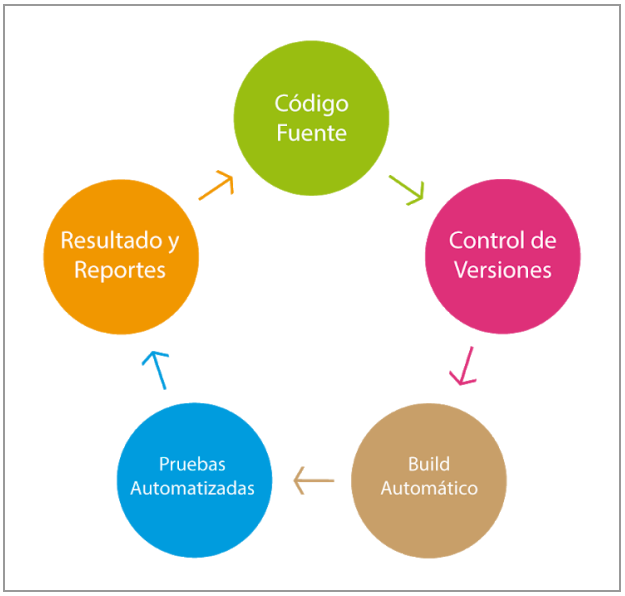


Imagen 70. Tareas en integración contínua.

Como podemos ver en la imagen 70, las herramientas de integración contínua constan de una serie de tareas configurables y actualmente hay muchas plataformas que agilizan el proceso de uso de la integración contínua como es el caso de TravisCI, que es un servicio que se conecta con un repositorio Github de forma rápida y a través de un fichero de configuración (imagen 69) se pueden personalizar los pasos de las tareas antes mencionadas.

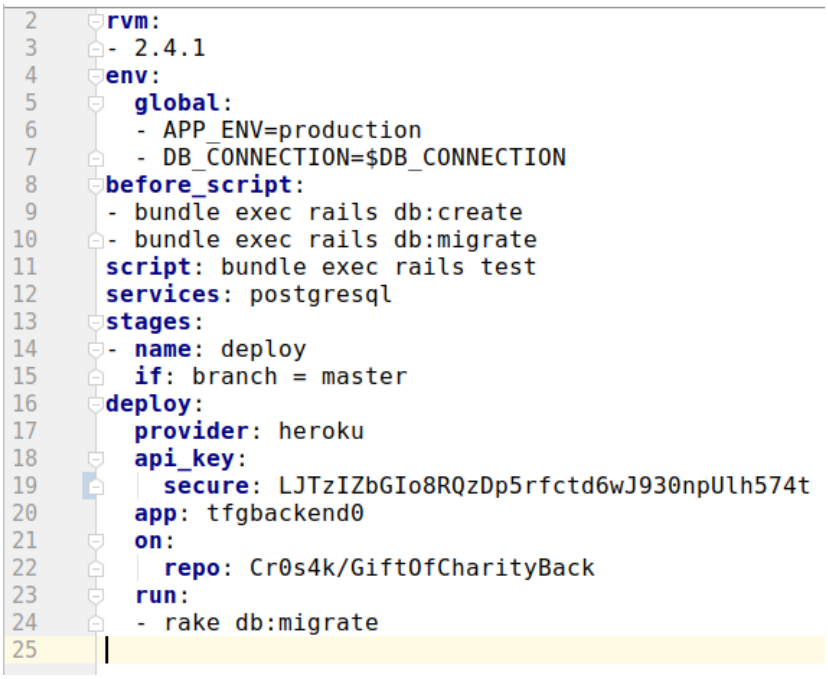


Imagen 71. Fichero de configuración para TravisCI del proyecto del servidor back-end.

Los ficheros de configuración de integración contínua definen las siguientes tareas:

1. Construcción del proyecto (build).
2. Ejecución de los tests (sólo en el proyecto del servidor back-end).
3. Comprobación de la rama del repositorio del código. Si la rama se llama ‘master’, es decir, la principal, se sube el código al servidor Heroku.

Pero además, como vemos en la imagen 71, a través del fichero de configuración también se puede definir la versión información sobre el compilador / intérprete a usar (líneas 2 - 3), variables de entorno que utiliza el proyecto (líneas 4 - 7), la configuración con el proyecto en Heroku para hacer un despliegue del proyecto (líneas 17 - 22).

10.2. Git Workflow

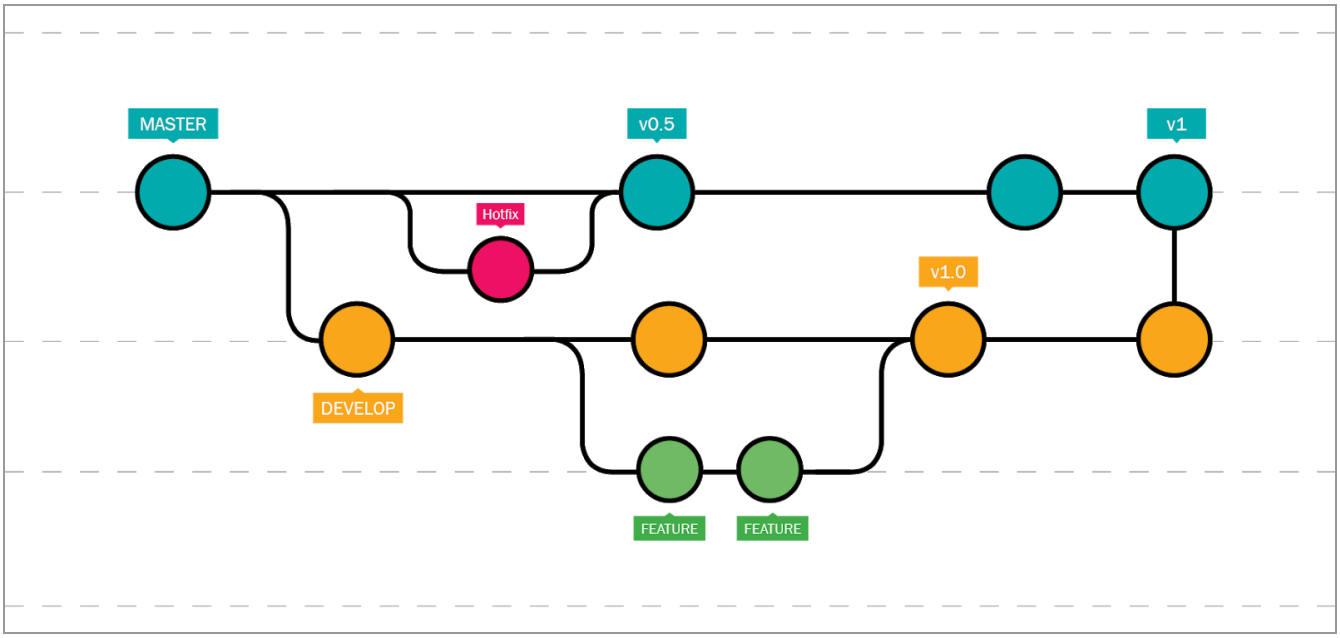


Imagen 72. Ramas en Git Workflow.

Git Workflow [45] sirve para tener un mejor control de versión de código, es una manera de trabajar en un proyecto a través de distintas ramas git[37]. La idea es que hay unas ramas primarias que son únicas y permanentes y otras secundarias que se crean a partir de las primarias y que su finalidad principal es acabar siendo fusionadas con las ramas primarias una vez el código esté listo.

En este flujo de trabajo las ramas principales son:
- **master**: es la rama pensada para ser producción, es decir, el código estable.
- **develop**: es la rama que contiene cambios y nuevas funcionalidades. Suele fusionarse con master una vez el código esté listo.

Además tenemos una serie de ramas secundarias que pueden no ser únicas (al contrario que master y develop):
- **feature**: son ramas que contienen nuevas funcionalidades. Las ramas feature suelen no estar relacionadas.
- **bugfix**: Sirven para corregir errores en la rama 'develop', ya que ésta se ha de evitar tocar directamente.
- **hotfix**: Similar a 'bugfix', pero éstas sirven para corregir errores en la rama 'master'.

Al principio del proyecto me encargué de subir el código inicial a 'master' y a partir de ésta rama crear 'develop'. Seguidamente, fui creando ramas 'feature' que contenían solamente la funcionalidad de una

[37] Bifurcaciones de código en un proyecto. Cada rama deriva de otra y éstas suelen usarse para realizar cambios sin alterar la rama a partir de la cual ha sido derivada. Las ramas pueden fusionarse con otras.

historia de usuario. Normalmente trabajaba con una sola rama feature y cuando ésta tenía código funcional, lo fusionaba con la rama ‘develop’.

Al cabo de cada dos semanas fusionaba la rama develop a master, ya que me encargaba de que el código estuviese preparado y así con successivamente con las siguientes iteraciones.

Si encontraba algún error en la rama ‘develop’, creaba una rama ‘bugfix’ para solucionarlo y lo fusionaba con ‘develop’. Y de la misma manera, si encontraba algún error en la rama ‘master’, creaba una rama ‘hotfix’, solucionaba el problema y la fusionaba a ‘master’.

Gracias a Git workflow he podido ir desarrollando el proyecto de manera gradual, teniendo un buen control sobre las diferentes partes del proyecto. Además, también tenemos la integración contínua con Travis CI[38], que como comenté en la sección anterior: a partir de un fichero de configuración, he establecido que si los cambios están situado en la rama 'master', que és única, se haga una actualización en Heroku, de manera que se suba el nuevo código estable.

10.3. Herramientas de seguimiento

10.3.1. PlanITpoker

En el capítulo de metodología menciono el uso PlanITpoker para estimar los puntos de las historias de usuario. Esta web proporciona una interfaz a partir de la cual se pueden introducir historias de usuario y una vez están puestas todas, permite invitar a una tercera persona con tal de ir valorando cada historia de usuario una por una según una escala de puntos: 0-1-2-3-5-8.

Con tal de tener una mejor visión de proyecto y ver de forma general si mi interpretación de la dificultad de las funcionalidades era correcta, decidí añadir a un miembro de la empresa para que pudiésemos estimar el proyecto y finalmente descubrir la causa de las diferencias entre los puntajes.

[38] https://travis-ci.org/

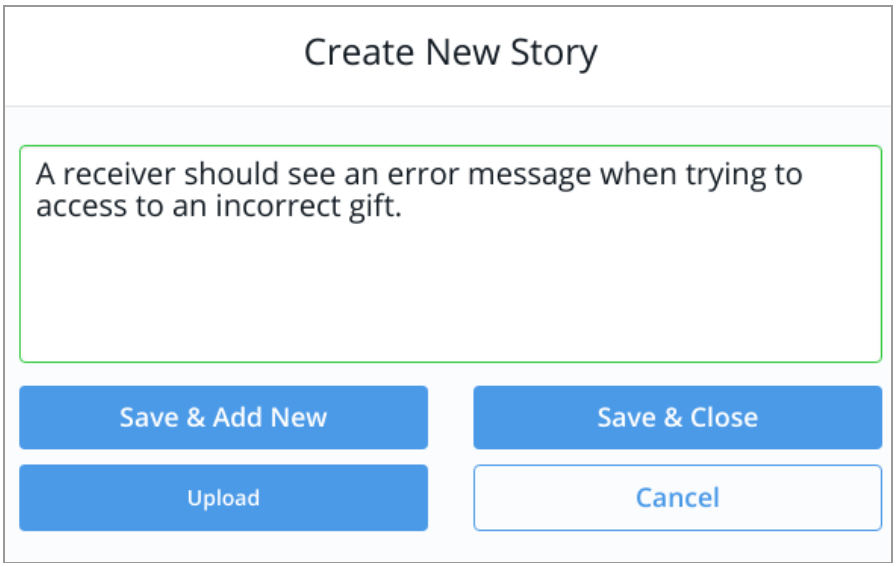


Imagen 73. Creación de una historia en PlanITpoker.

Como vemos en la imagen 73, PlanITpoker dispone de una interfaz minimalista que permite crear historias e ir añadiéndolas en una lista.

Imagen 74. Valoración de una historia en PlanITpoker.

Una vez tenía la lista de historias, fuí valorando historia por historia cada una de ellas, comentando el por qué de la puntuación puesta con mi compañero de la oficina.

A user can see a specific page for charity project available	1	00:01:58	
A user can see a list of charity projects	2	00:00:22	
An administrator can read information about a donation	3	00:01:04	
An administrator can mark a donation as send	1	00:00:17	
An administrator can see all the donations	5	00:03:26	
A donor can see a demo of their donation	2	00:00:20	
A receiver can consume/play a message	2	00:00:38	
A donor can buy a charity project	8	00:00:51	

Imagen 75. Parte de la lista de historias de usuarios con puntos estimados en PlanITpoker.

Finalmente, ya tenía todas las historias de usuario valoradas de forma racional y listas para ser pasadas a PivotalTracker.

10.3.2. Pivotal Tracker

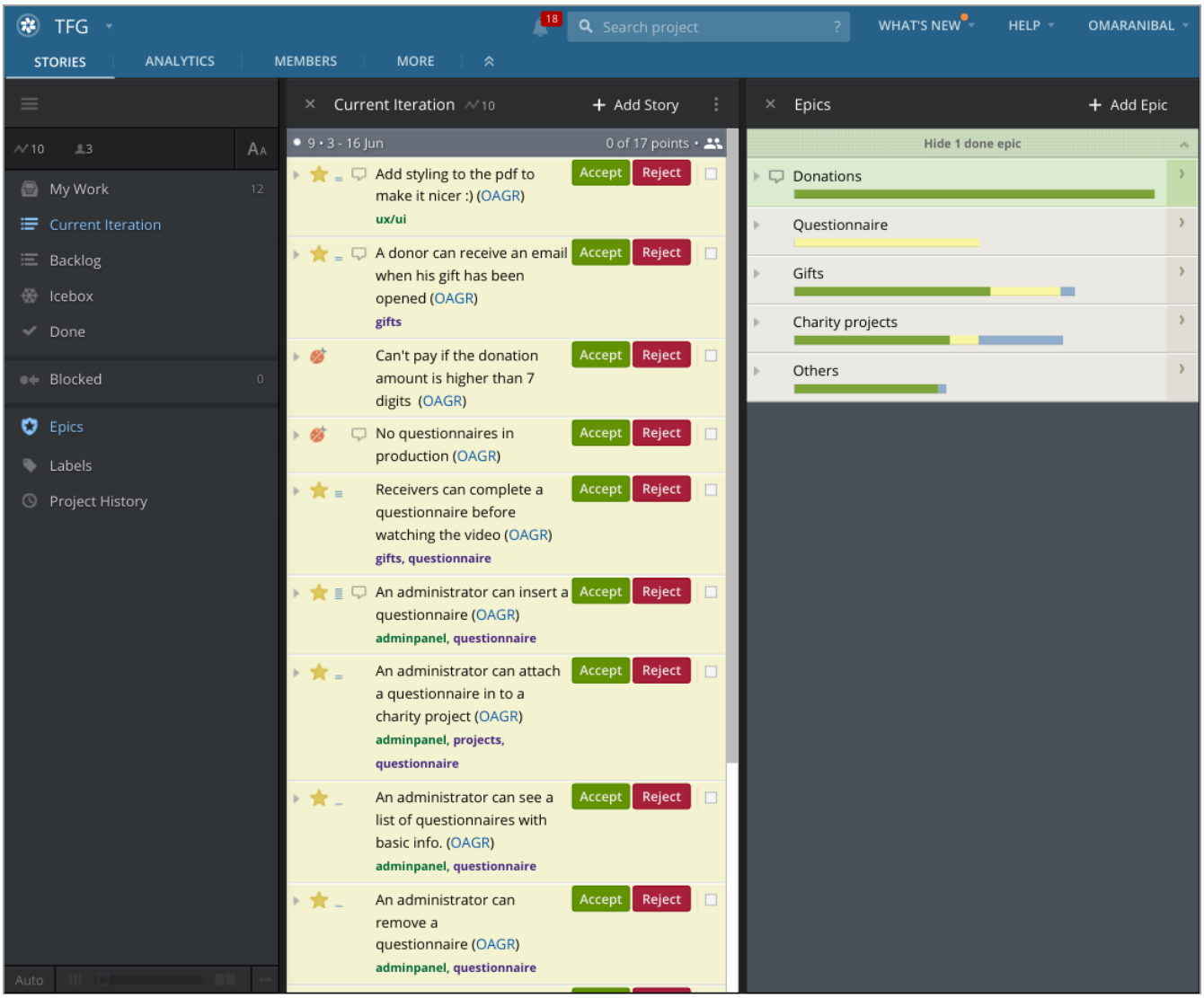


Imagen 76. Vista principal de Pivotal Tracker.

Pivotal Tracker ha sido una de las herramientas fundamentales durante el transcurso del proyecto. En esta aplicación web he añadido todas las historias de usuarios ya estimadas en una lista que actúa de contenedor de historias para cada iteración. Además, se han añadido errores y nuevas historias que han ido surgiendo de modo que toda la gestión del proyecto está en ésta aplicación web.

Como se ve en el apartado derecho de la imagen 76, las funcionalidades están divididas en épicas, tal y como lo están en la sección de historias de usuario del capítulo de requisitos del sistema. Lo bueno de Pivotal Tracker es que además de ser sencillo de usar, tiene muchas funcionalidades útiles como añadir etiquetas a cada historia, además de un apartado de comentarios donde se pueden incluir imágenes y demás con tal de comunicarte con la persona que lleva el proyecto (imagen 77).

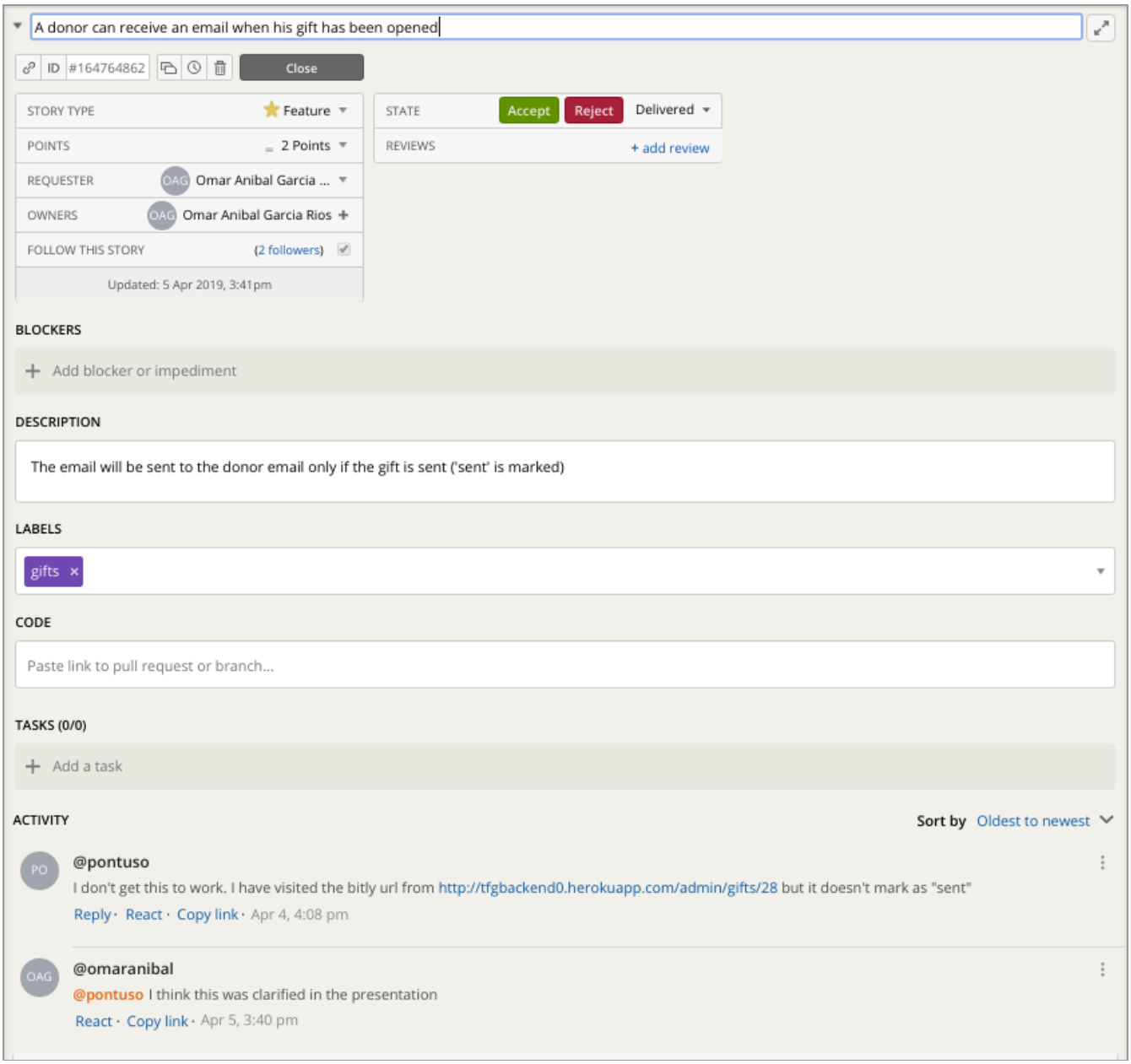


Imagen 77. Vista de una historia en Pivotal Tracker.

En mi caso, he ido completando cada historia de usuario y el director del proyecto ha ido aceptando o rechazando las historias de usuario, dejando comentarios dónde era necesario con tal de que corrigiese errores o detalles.

10.3.3. Github

Tanto el proyecto para el lado del cliente como el del lado del servidor, se ha usado Github para publicar el código. En Github hay diferentes planes y siendo estudiante se puede obtener un plan de

pago de forma que el proyecto sea privado, sin embargo se ha escogido el plan gratuito en ámbos proyectos y el código se encuentra de forma pública[39].

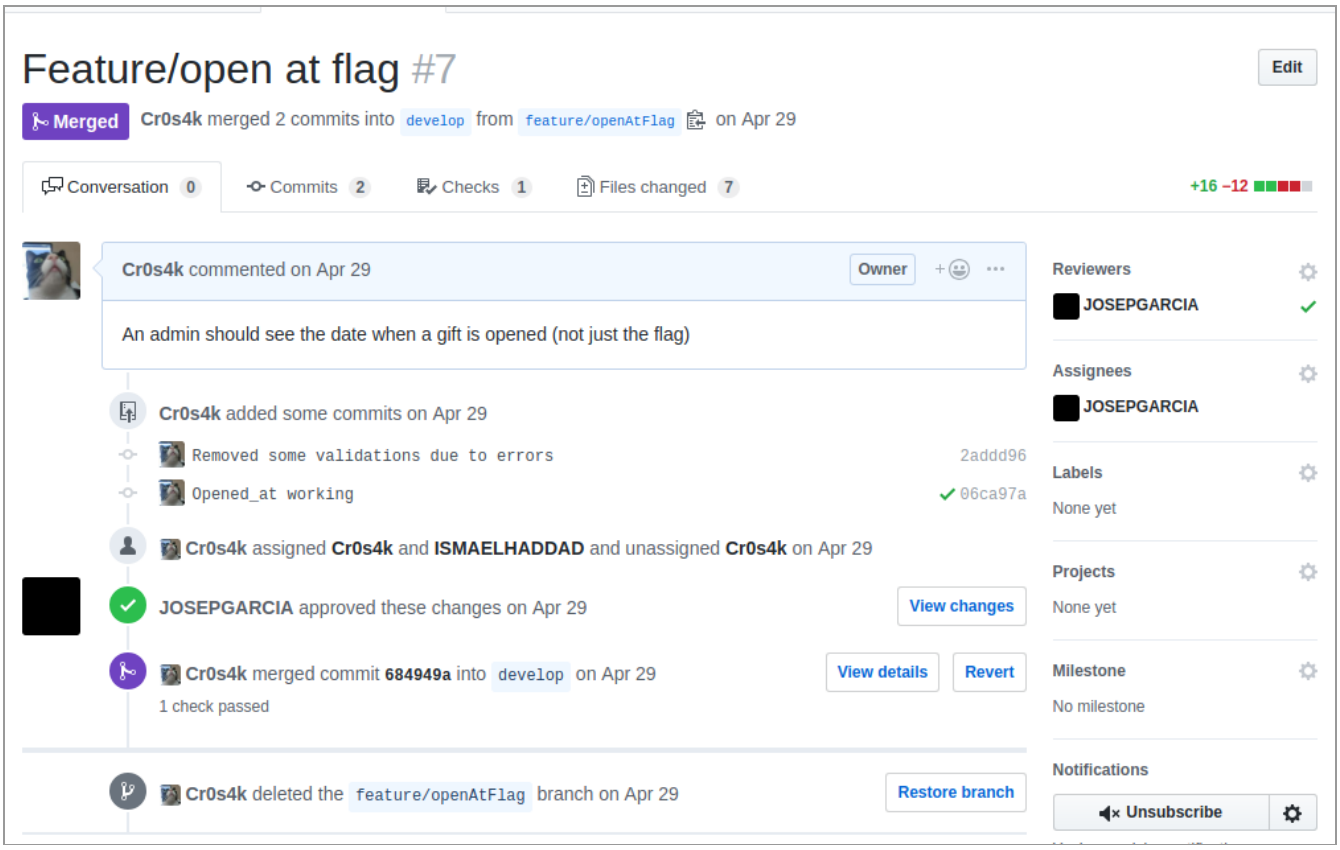


Imagen 78. Ejemplo vista principal de un pull request del proyecto del servidor back-end.

La principal funcionalidad de Github es alojar proyectos con el sistema de control de versiones Git [46], sin embargo hay otras funcionalidades muy útiles como por ejemplo los 'pull request'. Pongamos el caso de que hay una nueva pequeña funcionalidad que se quiera introducir en la rama 'develop', normalmente, en un equipo de desarrollo la persona encargada creará su rama y la subirá a Github. Posteriormente, para estar seguro de que el código es de calidad y no se han introducido problemas, puede haber un encargado o compañero que requiera que el desarrollador cree un 'pull request' que es una revisión de código donde se pueden poner comentarios y si finalmente el código es de calidad, fusionarlo con 'develop'.

En las últimas iteraciones, el director del proyecto me dijo que los pull request son una buena práctica y que a partir de ese momento las hiciera con un compañero de la empresa. Cada vez que acabara una funcionalidad, habría de crear un pull request para que revisaran mi código y si el código era aceptado,

[39] Proyecto del servidor back-end: https://github.com/Cr0s4k/GiftOfCharityBack (28 - 06 - 2019)
Proyecto del servidor front-end: https://github.com/Cr0s4k/GiftOfCharityFront (28 - 06 - 2019)

adjuntar el enlace en Pivotal Tracker, en la sección ‘código’ de las historias (la sección se puede ver en la imagen 77).

De forma resumida, podríamos decir que los pull request són una solicitud de revisión de código que alguien ha de aprobar para fusionarlo con otra rama. En mi caso, todos los ‘pull request’ són solicitudes para fusionar código de nuevas funcionalidades a la rama ‘develop’. En la imagen 78 podemos ver la vista principal que tiene un ‘pull request’, en panel de la imagen se pueden ver diversos apartados como ‘commits’, para revisar las porciones de código subidas al repositorio una por una, ‘checks’ que es la comprobación del código por integración contínua (TravisCI) y ‘files changed’ que son todos los ficheros cambiados, con su antes y después.

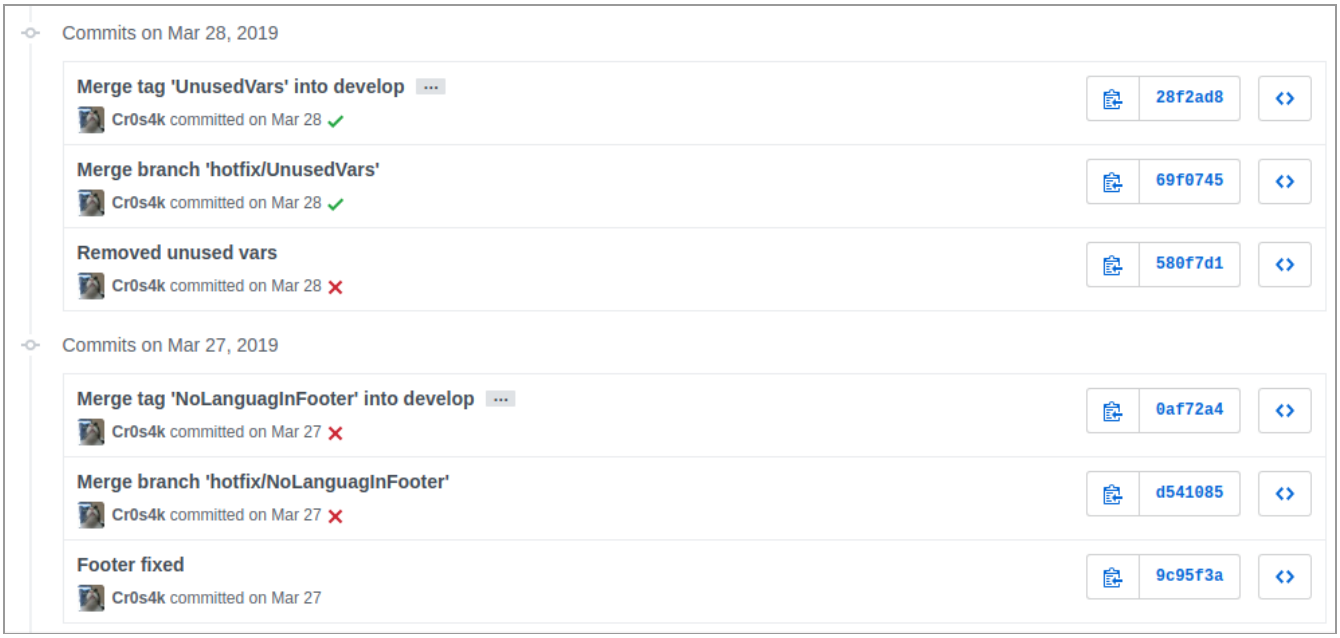


Imagen 79. Sección de ‘commits” en Github.

Finalmente comentar que github también me ha servido para integrar la integración contínua con TravisCI al proyecto, donde solo hay que buscar ‘integración contínua’ en la tienda de Github para agregar servicios y añadir a TravisCI. Además, una cosa que me ha gustado bastante de la integración de TravisCI en Github es que en la sección de ‘commits’, se añade un icono que indica el estado de la salida del proceso que realizar la integración contínua (imagen 79).

10.3.4. Heroku

A lo largo de este capítulo he ido mencionando Heroku, que es un servicio que ofrece la posibilidad de desplegar aplicaciones para que puedan ser accedidas de forma pública a través de un enlace. Lo bueno de Heroku es que es muy fácil de usar, ya que solo se ha de especificar las tecnologías del proyecto y Heroku lo configura todo para que cuando se suba el proyecto, se instalen las librerías del proyecto, se compile y demás [47].

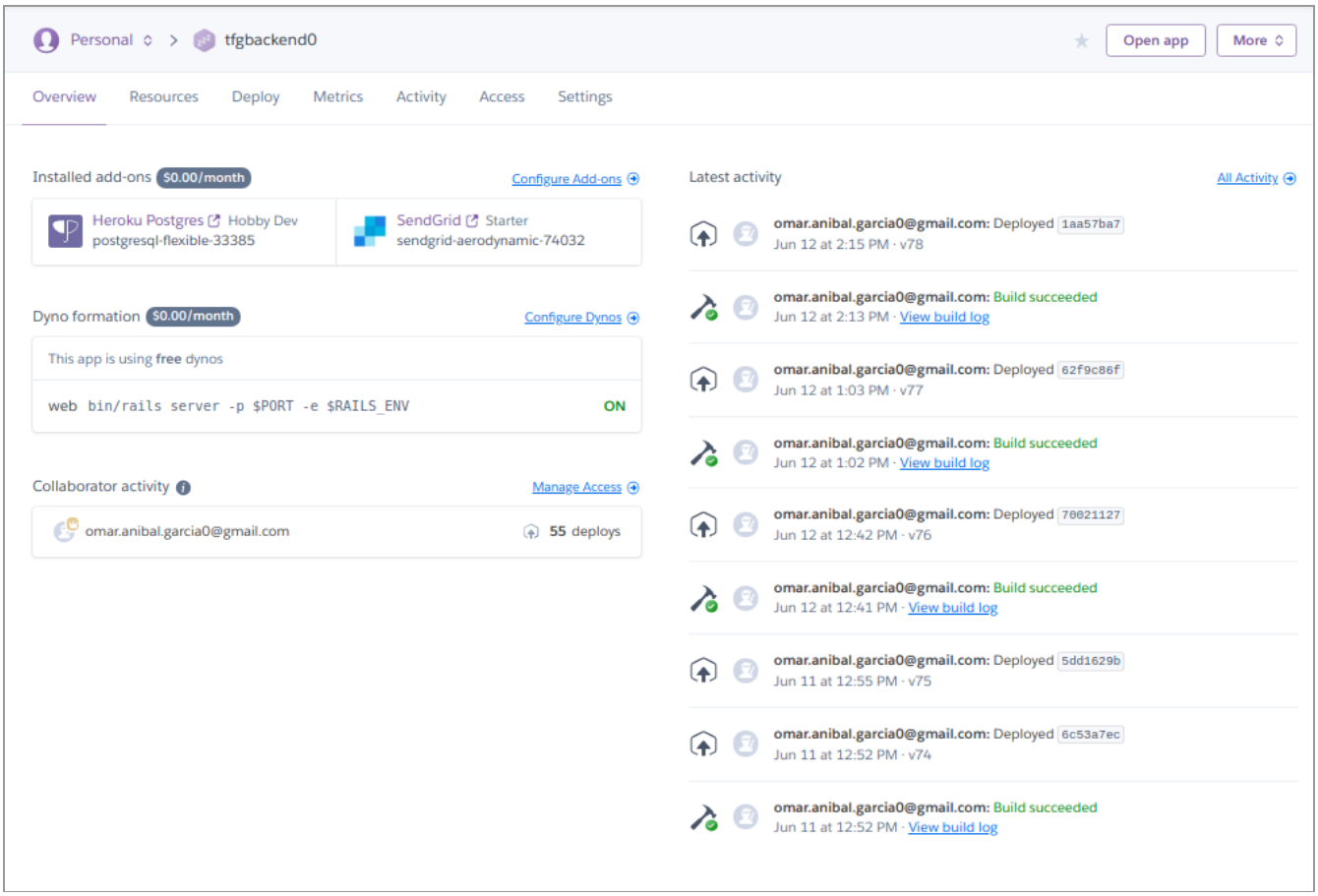


Imagen 80. Panel principal Heroku del proyecto back-end.

Para trabajar en Heroku, se ha de verlo como un repositorio git, en el que todo el código que se encuentre allí está subido en un servidor y puede ser accedido de forma pública. La configuración del proyecto front-end fue muy básica, ya que no contiene ninguna base de datos ni ejecutables externos, lo único que hice fue añadir las variables de entorno necesarias en el apartado de configuración.

Sin embargo, para el proyecto del servidor back-end tuve que añadir el complemento Heroku Postgres para obtener una base de datos y el complemento SendGrid, que sirve para poder enviar e-mails. Además, también tuve que añadir una dependencia externa llamada 'wkhtmltopdf', que es un ejecutable externo que necesita la librería que utilizo para poder generar pdf's. En la parte izquierda de la imagen 80 podemos ver los complementos de Heroku Postgres y SendGrid utilizados en el proyecto del servidor back-end.

Ambos proyectos usan el plan gratuito de Heroku que tiene 512MB de RAM y un proceso activo, que es más que suficiente para este TFG.

10.4. Proceso

Tal y como comenté en la el capítulo de metodología, el director del proyecto ha usado Pivotal Tracker y además se han ido haciendo reuniones semanales donde se hablaba del desarrollo del proyecto.

También comentar que también me he reunido con un desarrollador senior de la empresa para que revise mi código tal y como indica el capítulo de metodología de la memoria, aunque ésto no ha ocurrido cada semana, sino que ha sido cada dos o tres.

Además, durante el transcurso del proyecto, la idea ha sido centrarse solamente en lo importante del producto, de manera que si no me daba tiempo a tener todas las funcionalidades, al menos tendría algo funcional. Esta forma de llevar el proyecto ha sido útil para tener resultados pronto y además, me ha permitido centrarme en cosas interesantes. Por ejemplo, cuando estaba definiendo los requisitos, había una historia de usuario llamada inicio de sesión / cierre de sesión, que finalmente se ha eliminado pues realmente en este proyecto no era necesario, ya que como podemos comprobar, esta aplicación funciona bien sin inicio ni cierre de sesión.

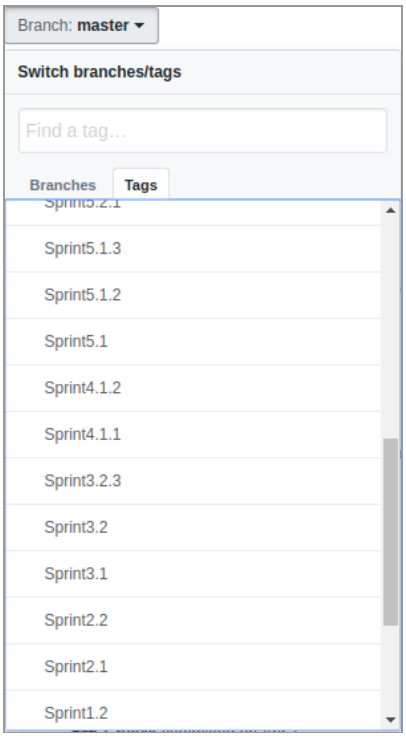


Imagen 81. Tags del proyecto del servidor front-end en Github.

Sobre la metodología de trabajo, he explicado antes en la sección de Git Workflow que la rama ‘master’ es la estable, es decir, producción. Para separar las iteraciones del proyecto, toda fusión de la rama ‘develop’ con ‘master’ corresponde a una iteración, esta fusión de ramas se suele llamar ‘release’ y con tal de que sea fácil de buscar en Github y tener aún una mayor separación, se han puesto añadido unos ‘tags’ para que desde Github, se pueda seleccionar la release correspondiente a la iteración que queramos, tal y como se ve en la imagen 81 y así ver el estado del código en cada release.

Respecto a la memoria, comenté en el capítulo de metodología que la documentación del proyecto se presentaría en forma de metodología en cascada. Como se puede comprobar, tenemos el capítulo de especificación, diseño e implementación. Éstos se han ido escribiendo a lo largo del proyecto y se han

reescrito y reorganizado, de tal manera que así se presenta solo los artefactos finales de cada etapa para que quede todo más compacto y claro.

Finalmente, he de decir que he utilizado herramientas usadas en proyectos de metodología ágil y estoy contento de cómo ha ido evolucionando el proyecto de forma totalmente controlada. Obviamente ésto es más fácil cuando solo es un desarrollador, pero estoy convencido de que la metodología explicada en esta sección se puede aplicar en proyectos ágiles con más personas de forma que sigue resultando eficaz.

11. Testing

En todo software serio se han de realizar pruebas para asegurar la calidad del producto. El hecho de llevar un proyecto sin tests conlleva a tener una posibilidad muy alta de tener errores y que éstos se vayan acumulando a lo largo del proyecto. Es por ello que en este proyecto se han llevado a cabo dos tipos de tests: automáticos y manuales. Los tests automáticos se han utilizado para el proyecto del servidor back-end, mientras que los manuales, se han utilizado para ámbos proyectos, back-end y front-end.

11.1. TDD

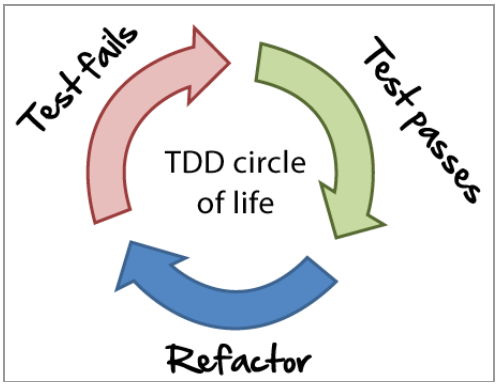


Imagen 82. Diagrama del proceso de testing TDD.

En la primera iteración surgió la idea de realizar 'Test-Driven-Development', que consiste en escribir primero los tests, después escribir el código que pase la prueba y refactorizar el código. Así sucesivamente hasta conseguir la funcionalidad deseada con las comprobaciones esperadas [48]. El hecho de hacer TDD tiene cosas positivas como un código más seguro, robusto, mantenible y se enfoca más en las necesidades, ya que primero primero se analiza lo que se necesita que el código haga y no al revés.

Sin embargo, TDD suele tener una curva de aprendizaje no muy sencilla y cuesta acostumbrarse. Pensé en aplicar TDD en la primera iteración, pero al no haberlo aplicado antes en un proyecto 'serio' noté una productividad baja y no me acabé de acostumbrar, por lo que decidí no aplicar TDD y realizar tests una vez el código está escrito.

11.2. Tests automáticos

Los tests automáticos son las que son ejecutadas por TravisCI cada vez que se sube código al repositorio Github. Estas pruebas se han escrito para el servidor back-end con una librería incluida en Ruby on Rails llamada 'minitest' para hacer tests unitarios[40].

[40] Test para comprobar una porción de código.

En el proyecto del servidor back-end se han realizado dos tipos de pruebas automáticas. Por un lado, tenemos los tests de modelo, que miran de comprobar las asociaciones y validaciones de los modelos. Mientras que por otro lado, tenemos los tests funcionales, que se centra en las funciones de los controladores de la API.

11.2.1. Tests de modelo

```
1   require 'test_helper'
2
3   class QuestionTest < ActiveSupport::TestCase
4     # Associations
5     should have_many(:answers).dependent(:destroy)
6     should belong_to(:questionnaire).counter_cache(:question_count)
7
8     # Validations
9     should validate_presence_of(:text)
10  end
11
```

Imagen 83. Test de modelo ‘question’.

Para cada modelo del proyecto se han hecho tests para comprobar las asociaciones y validaciones de los atributos. Por ejemplo, en la imagen 83 vemos el test del modelo ‘question’, que si recordamos el esquema de la base de datos, tiene muchas preguntas asociadas (answers). Por lo que en la línea 5 vemos que se comprueba que tenga preguntas y que además, si se borra la pregunta en sí, se borren las respuestas, es decir, que haya una dependencia al destruir las preguntas, ya que no tiene sentido que existan respuestas para una pregunta que no está en la base de datos.

Los tests escritos son muy intuitivos y fáciles de escribir gracias a una librería llamada Shoulda Matchers [49]. Por ejemplo, en la línea 6 vemos que se comprueba que las preguntas deben corresponder a un cuestionario y que hay un contador llamado ‘question_count’ en el cuestionario que hace referencia a las preguntas que pertenecen al cuestionario. También en la línea 9 se indica que tiene que existir un texto (text) para la pregunta.

```
Work directory: /home/cr0s4k/Projects/GiftOfCharityBack
Loading files....
========================================
1. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/questionnaire_test.rb:1
2. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/gift_test.rb:1
3. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/donor_test.rb:1
4. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/charity_project_test.rb:1
5. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/admin_user_test.rb:1
6. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/answer_test.rb:1
7. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/receiver_test.rb:1
8. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/donation_test.rb:1
9. /home/cr0s4k/Projects/GiftOfCharityBack/test/models/question_test.rb:1

9 files were loaded.
========================================
Running tests...
Started

Finished in 0.18744s
29 tests, 29 assertions, 0 failures, 0 errors, 0 skips

Process finished with exit code 0
```

Imagen 84. Ejecución de los tests de modelo.

Como vemos en la imagen 84, en total tenemos 29 tests para todos los modelos y todos están pasados satisfactoriamente.

11.2.2. Tests funcionales

Los tests funcionales son tests que se han creado para probar la funcionalidad de las funciones de los controladores, es decir, de las rutas de la API.

Controlador proyectos de caridad:

- GET /charity_projects
    - Se ha comprobado que la respuesta es un 200 (correcto).
    - Se ha comprobado el formato de la respuesta.
    - Se ha comprobado que devuelve todos los proyectos de caridad habilitados.
- GET /charity_projects/:id
    - Se ha comprobado que la respuesta es un 200 con un identificador correcto (:id).
    - Se ha comprobado que el proyecto de caridad obtenido a partir de un identificador existente tiene un formato de salida y valores correctos.
    - Se ha comprobado que se devuelve un error cuando se solicita un proyecto de caridad con un identificador inexistente o incorrecto en la base de datos.

Controlador de donaciones:

- POST /donations

- Se ha comprobado que se obtiene un estado 200 (correcto) en realizar una donación con todos los parámetros correctos.
- Se ha comprobado que se obtiene un estado erróneo cuando se realiza una donación sin algún parámetro requerido.

Controlador de regalos:

- GET /gifts
  - Se ha comprobado que el regalo obtenido a partir de un identificador existente tiene un formato de salida y valores correctos.
  - Se ha comprobado que se obtiene un estado 200 (correcto) en realizar una petición con un token correcto.
  - Se ha comprobado que se obtiene un error en realizar una petición de un regalo con un token incorrecto o inexistente.

Controlador de cuestionarios:

- GET /questionnaires/:id
  - Se ha comprobado que la respuesta es un 200 con un identificador correcto (:id).
  - Se ha comprobado que el cuestionario obtenido a partir de un identificador existente tiene un formato de salida y valores correctos.
  - Se ha comprobado que se devuelve un error cuando se solicita un cuestionario con un identificador inexistente o incorrecto en la base de datos.

```
10  test name "should make a donation" do
11    order_id = make_order( amount 5)
12    post donations_url(
13       orderId: order_id,
14       itemId: CharityProject.first.id,
15       videoUrl: 'http://somevideo.com',
16       address: 'TestAddress',
17       city: 'Barcelona',
18       country: 'Spain',
19       province: 'Barcelona',
20       postcode: '08202',
21       email: 'test@test.com',
22       amount: '5',
23       donorName: 'Donor'
24     )
25    assert_response type 200
26  end
```

Imagen 85. Ejemplo de test funcional en POST /donations.

En la imagen 85 podemos ver un ejemplo de test funcional en la ruta /donations con el método POST. En este test se comprueba que se obtenga un 200 (correcto) en la salida. Si miramos la línea 11, se hace una llamada a 'make_order', que es una función que se conecta con Paypal para simular un pago y obtener un identificador de compra, ya que en la acción de realizar donación se comprueba con el servidor de Paypal que se pase un identificador correcto, de manera que fue necesario simular una compra.

Finalmente, en la imagen 86, vemos la salida de la ejecución de todos los tests funcionales. En total tenemos 15 tests con 43 comprobaciones y 0 fallos.

```
Work directory: /home/cr0s4k/Projects/GiftOfCharityBack
Loading files....
=========================================
1. /home/cr0s4k/Projects/GiftOfCharityBack/test/controllers/gifts_controller_test.rb:1
2. /home/cr0s4k/Projects/GiftOfCharityBack/test/controllers/charity_projects_controller_test.rb:1
3. /home/cr0s4k/Projects/GiftOfCharityBack/test/controllers/questionnaires_controller_test.rb:1
4. /home/cr0s4k/Projects/GiftOfCharityBack/test/controllers/donations_controller_test.rb:1

4 files were loaded.
=========================================
Running tests...
Started

Finished in 5.32508s
15 tests, 43 assertions, 0 failures, 0 errors, 0 skips

Process finished with exit code 0
```

Imagen 86. Ejecución de los tests funcionales.

11.3. Tests manuales

Respecto a los tests manuales, han estado presentes tanto en el proyecto del servidor back-end como en el del servidor front-end. En el primero, se ha usado Postman para realizar llamadas a la API, además probar funcionalidades del panel de administración de forma directa, mientras que para el segundo me he centrado en la experiencia de usuario y diseño del sistema, donde gracias a la ayuda de terceras personas que obtenido reacciones para hacerlo lo mejor posible.

11.3.1. Postman

Durante el desarrollo del proyecto del servidor back-end se ha desarrollado un API. Esta API se puede probar con tests unitarios, pero a veces es necesario hacer peticiones a la API de manera rápida, por lo que utilizar una herramienta para analizar llamadas HTTP resulta muy útil.

En la imagen de abajo podemos de uso a la ruta /donations con Postman. En la parte superior vemos la ruta especificada donde se puede elegir el tipo de método HTTP y además especificar parámetros. Después en la parte inferior se ven diversas pestañas donde cada una de ellas permite ver todos los

detalles de la petición HTTP. En este caso, la pestaña 'body' está seleccionada y vemos que muestra un error con estado 400 que dice que falta el parámetro 'orderId' en la llamada.

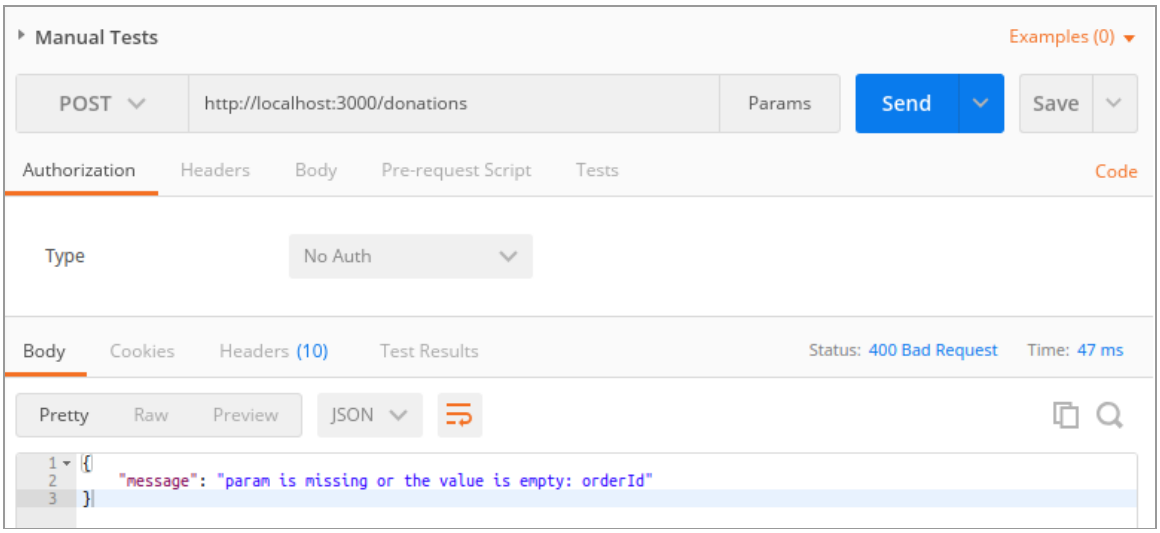


Imagen 87. Ejemplo de HTTP POST con Postman a /donations.

Esta forma de realizar tests no se ha hecho en todas las rutas de la API, sino en las que más complejas eran y sólo cuando era necesario.

11.3.2. Panel de administración

Para las funcionalidades del panel de administrador se han realizado tests manuales debido a que la librería de ActiveAdmin consta ya de tests automáticos internos. De tal manera que para agilizar el proceso de desarrollo consideré que no era necesario realizar tests. Sin embargo, para asegurar el correcto funcionamiento del panel, he comprobado las funcionalidades de manera manual.

11.3.3. Diseño de interfaces y experiencia de usuario (UI & UX)

Otro tipo de tests han sido los manuales tocando directamente la aplicación. En este caso, con cada historia de usuario he ido comprobando el diseño de la interfaz y experiencia de usuario. Sin embargo, no solo la he probado yo, sino que la aplicación web ha sido probada por mis compañeros de oficina, el director de la empresa y finalmente, por mi entorno más cercano ajeno al proyecto.

Con cada reunión con mi director, éste nos ofrecía sus comentarios acerca del diseño y la experiencia de usuario. De hecho, ya comenté en la sección de Material UI en el capítulo de implementación que a mi director, al principio el diseño no le resultaba atractivo. Es por ello que decidí coger la base de una plantilla para desarrollar el proyecto con un diseño mucho más profesional y ampliarlo a mi modo.

Además, he ido enseñando el proyecto a algunos compañeros de oficina para que me dieran su visto bueno y creo que el proyecto ha ido yendo por buen camino gracias a comentarios constructivos. También decir que al final de la quinta iteración mi director me propuso hacer una pequeña

presentación del proyecto para toda la oficina y así también recibir comentarios de las personas que no habían visto todo el proyecto.

También le he enseñado el proyecto a mi entorno más cercano con tal de ver si eran capaces de realizar una donación por ellos mismos y además entender el funcionamiento de la aplicación. Ya que para mi, que soy el único desarrollador, me resulta muy fácil e intuitivo precisamente por la razón de que he sido yo quien ha construido el proyecto y a veces es difícil detectar los propios fallos.

12. Sostenibilidad

12.1. Autoevaluación

En estos años en la FIB, la sostenibilidad nunca ha tenido cabida de forma principal en ninguno de mis proyectos o trabajos anteriores. Antes de empezar a realizar el TFG solía llevar rienda suelta a mis ideas centrándome sólo en el núcleo de lo que hacía, ignorando partes que en ese entonces no me resultaban importantes. No obstante, siempre he trabajado basándome en una documentación sólida, en otros otros proyectos, ideas y personas, por lo que finalmente he acabado realizando trabajos sostenibles sin yo darme cuenta de ello. Además, gracias a la enseñanza uno siempre (al menos en mi entorno) tiene en cuenta matices tales como el medio ambiente debido a los diversos documentales que se suelen enseñar en clase, aunque de forma muy superficial. Y no solo eso, debido a la cultura presente hoy en día, también es normal tener ciertos conocimientos sobre por ejemplo el impacto social que causa la salida de nuevas tecnologías y cómo ésto afecta económicamente a la población.

De todos modos, después de leer los diversos documentos y ejemplos proporcionados por el curso GEP, puedo decir que ahora sí soy capaz de comprender y tener en cuenta la sostenibilidad de un proyecto TIC de forma consciente. De hecho, es lo que voy a demostrar a través de las siguientes secciones de este documento. Además, he tenido en cuenta que debido a que uno de los puntos de mi proyecto se centra en la mejora del medio ambiente, es útil analizar el producto en la dimensión de sostenibilidad ambiental, y estoy muy seguro de que ha marcado mi trabajo de los últimos meses, ya que este análisis me ha permitido aumentar la sostenibilidad de este trabajo de fin de grado.

12.2. Dimensión económica

Respecto a la dimensión económica, se ha abarcado el tema con profundidad y eso se vé reflejado en el capítulo de gestión económica, donde se ha realizado un presupuesto a partir de diversos costes donde también se han tenido en cuenta posibles imprevistos que pudiesen surgir durante todo el desarrollo del proyecto.

Además, también se ha comentado sobre los gastos económicos que puedan surgir una vez realizado el proyecto, es decir, durante la vida útil del software. Sin embargo, no se han entrado en muchos detalles pues este trabajo se centra en la construcción de un software, pero aún así, es útil tener una idea del futuro del proyecto.

12.3. Dimensión ambiental

En este proyecto, el tema ambiental toma un fuerte papel. De forma general, el proyecto busca crear un espacio donde se realicen donaciones y que éstas acaben formando parte en la contribución de un proyecto dedicado al medio ambiente. Aunque no hemos de olvidar que estamos hablando de un prototipo, donde una de las ideas es que el dinero que se obtendría a través de donaciones se

repartiese realmente a proyectos ambientales. De hecho, una de las cosas que me ha comentado la empresa es que busque este tipos de proyectos y que encontrase una manera de hacer que los pagos de las donaciones fuesen a parar a estos proyectos, sin un intermediario de por medio. Sin embargo, finalmente decidí proponer una solución general a ésto y centrarme más en otros aspectos más tecnológicos.

Este TFG también tiene la finalidad de intentar minimizar las compras innecesarias que se hace como regalo, introduciendo una experiencia puramente virtual a través de la aplicación web. Es por ello que pienso que este es un buen punto a favor respecto a la dimensión ambiental del proyecto. De todos modos, si bien la intención es que los regalos no sean cosas materiales, la forma de recibir el regalo sí lo será. La carta como método de envío de regalo es algo que se ha pensado de quitar a cambio de un SMS[41] con un link a la aplicación, pero junto al director de TFG hemos pensado que eso quitaría cierta magia al producto, por lo que este sería un pequeño punto en contra: el gasto del papel y los derivados del transporte de carta hacia la dirección del receptor del regalo.

Respecto a los recursos, el proyecto se realizará en mi ordenador portátil personal, por lo que he reutilizado este recurso. Sobre la empresa, se hace uso de internet y del aire acondicionado, que normalmente consume mucha electricidad, pero al no ser yo la única persona que consume estos recursos, no tendrá mucho impacto en la sostenibilidad ambiental del proyecto.

Por otro lado, no he de olvidar los gastos derivados de los servidores de los servicios de terceros que utiliza mi aplicación. Esto es muy difícil de calcular, aunque lo positivo de esto es que al estar usando planes que no son de pago, a las empresas que gestionan estos servicios de terceros no les interesa pagar más de lo que uno necesita, por lo que en este punto no veo que haya un gran impacto ambiental, pues mi consumo será ínfimo.

Como conclusión, puedo decir que el proyecto tiene una buena puntuación respecto a la sostenibilidad ambiental. Aunque pienso que siempre hay algo con lo que mejorar, en este caso quizás sería el quitar el envío de cartas, pero eso sería restar valor al producto, por la cual cosa se ha decidido no prescindir de ello. Además, mi solución es ambientalmente mejor que las soluciones existentes pues como ví en el estudio de mercado, normalmente se suelen enviar postales con imagenes y demás, lo cual gasta tinta, pero en mi proyecto se envía un pequeño texto con un enlace escrito sin colores muy llamativos, reduciendo el consumo de recursos al mínimo.

12.4. Dimensión social

Respecto a la dimensión social, he de decir que a nivel personal este proyecto es una gran fuente de conocimiento técnico. He utilizado tecnologías que nunca había utilizado antes, además de aprender a construir un proyecto de software tomando todos los roles que hay en él (o al menos la mayoría).

[41] Mensaje corto de texto que se envía a través de de la telefonía móvil.

Pienso que este proyecto va a tener un gran impacto en mi futuro laboral a corto plazo, ya que además es mi primera vez trabajando en una empresa haciendo ocho horas diarias.

Haciendo referencia al contexto del proyecto, este TFG solucionará el problema de tener que cargar a los receptores de regalos con cosas no útiles, pero además de ello, este trabajo es una buena solución para aquellas personas (en este caso los donantes) que no saben qué regalar a personas en situaciones donde es tradición dar algo. Por lo que pienso este proyecto tiene un buen impacto social. Además, la calidad de vida social se verá mejorada pues los usuarios receptores de regalos se llevarán una grata sorpresa y no sólo eso, sino que además estamos concienciando sobre los problemas actuales ambientales a la población.

Como conclusión, quiero añadir que pienso que la aplicación podría solucionar los pequeños problemas sociales antes dichos, así que sí existe una necesidad real del proyecto. Puede que no toda la sociedad necesite el producto en el que trabajaré, pero sí una pequeña parte.

12.4. Matriz de sostenibilidad

Para acabar, en la tabla 54 se muestra la matriz de sostenibilidad del proyecto teniendo en cuenta el primer bloque 'Proyecto puesto en producción' del que se ha hecho una la valoración económica, ambiental y social.

Sostenibilidad	**Económica**	**Social**	**Ambiental**	**Total**
Proyecto puesto en producción (PPP)	Factura	Impacto personal	Consumo de diseño	
	9	10	9	**28 / 30**

Tabla 54. Matriz de sostenibilidad

13. Identificación de leyes y regulaciones

En el capítulo de requisitos no funcionales de la memoria tengo el requisito de cumplimiento legal. Debido a que lo que se intenta es realizar un sistema serio, éste ha de estar al tanto de las leyes actuales. Sin embargo, éste tema se tocará de forma superficial sin entrar en muchos detalles por lo que he decidido acotar el alcance de éste apartado en el trabajo.

13.1. Datos de usuarios

La ley de protección de datos establecida en 1999 [50] vela por la seguridad de la información de los usuarios en Internet. En este proyecto, se ha añadido la funcionalidad de poder borrar los datos de los usuarios desde el panel de administración con tal de mantenerse al margen de la ley. Además, la información muy confidencial como las contraseñas, se mantienen encriptadas en la base de datos.

13.2. Librerías

Las librerías utilizadas en este proyecto son de npm[42] para el proyecto front-end y rubygems[43] para el proyecto back-end. Ambos sitios web almacenan librerías de código abierto de otros desarrolladores por versiones que pueden ser referenciadas en los proyectos para instalarlos por comandos sin tener que estar descargando las librerías manualmente desde las webs antes dichas.

En ambos proyecto se han utilizado solamente librerías de código abierto, sin embargo hay diferentes tipos de licencias que tienen las librerías que he usado que son [51]:

- Licencias copyleft fuerte: Exigen la publicación de las obras con el código fuente, que el trabajo derivado del original conserve la misma licencia.
- Licencias permisivas: Son licencias flexibles que permiten que la obra pueda ser redistribuida como libre o privada.

13.3. Cookies

Por último, mencionar que debido a que no se utilizan cookies[44] en la aplicación, no se mostrará un aviso de alerta como indica el criterio de satisfacción del requisito no funcional de legalidad.

[42] https://www.npmjs.com/
[43] https://rubygems.org/
[44] Es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de esta forma, el sitio web puede consultar ésta información previa del navegador. [52]

14. Conclusiones

14.1. Planificación final

Partiendo de la planificación inicial que era la siguiente (600h):

1. Puesta en marcha (80 h): Configuración de entorno, aprendizaje tecnologías, definición de requisitos.

2. Gestión del proyecto (75 h): Asignatura GEP.

3. Desarrollo de la aplicación 1 (55h x 3 iteraciones = 165h): Desarrollo de la aplicación cursando la asignatura GEP.

4. Desarrollo de la aplicación 2 (80h x 3 iteraciones = 240h): Desarrollo de la aplicación únicamente.

5. Documentación y presentación (40h): Redacción memoria y presentación oral.

Han habido variaciones en el punto 3 (Desarrollo de la aplicación 2) de la planificación inicial debido a un mal cálculo de los días festivos (semana santa). Si recordamos el diagrama de Gantt inicial, hay un espacio de una semana entre la iteración 4 y 5, este espacio fue un error, pues al tener un contrato en empresa, no tenía fiesta de toda una semana, sino que fueron 2 días (19 Y 22 de abril), por lo que decidí cambiar la planificación desplazando la iteración 5 y 6 una semana atrás.

Además, la última iteración del desarrollo no ha ocurrido tal y como está planeada. En esta última iteración ya tenía todas las funcionalidades indicadas en las historias de usuario hechas, sin embargo, habían partes del código sin testear aún, por lo que dediqué todo tiempo al testing y documentación del proyecto.

También he tenido que participar en un proyecto de la empresa, que ha hecho que le dedique la parte de documentación y presentación se desplacen hacia delante y le dedique menos tiempo. Cabe decir que también tuve que hacer unos retoques a la aplicación, por lo que también tuve que dedicarle tiempo.

Teniendo el cuenta estos cambios, he realizado un último diagrama de Gantt:

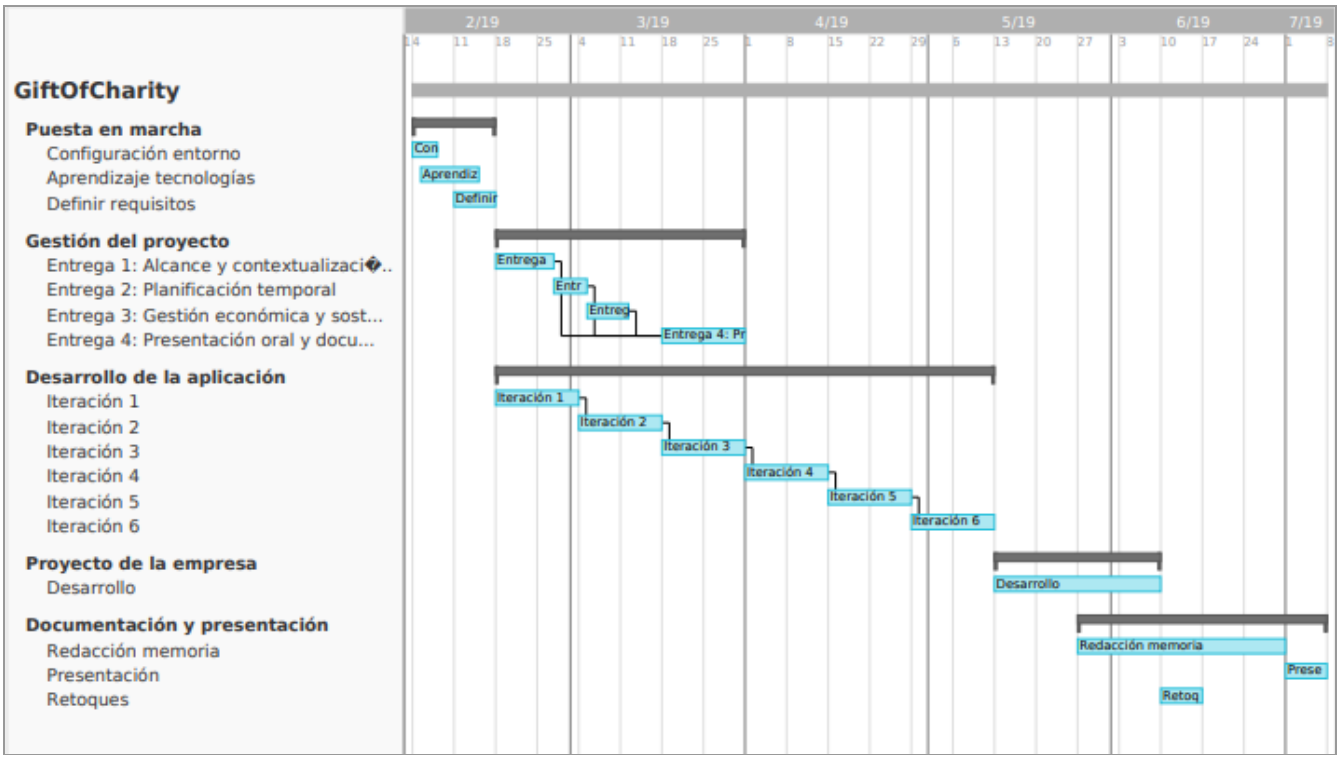


Imagen 88. Diagrama de Gantt final.

Como podemos ver en el diagrama de Gantt, la etapa de desarrollo duró 4 semanas todo trabajaba casi exclusivamente en el proyecto de la empresa. En total hice 150 horas, ya que en las últimas dos semanas dediqué una hora cada día en la memoria del proyecto e informe de seguimiento.

Inception	80
Sprint 1	80
Sprint 2	80
Sprint 3	80
Sprint 4	80
Sprint 5	64
Sprint 6	72
Total	536

Imagen 89. Excel que indica las horas del desarrollo de la aplicación.

Tengo un excel donde he apuntado las horas hechas diariamente durante todas las iteraciones y como podemos ver, se han hecho 536 horas de desarrollo con la iteración 6 incluida. A esto le hemos de sumar las 10 horas dedicadas al TFG durante mi desarrollo del proyecto de la empresa y la última semana de mi estancia en la empresa en la que le dedique todas las horas a la memoria y a los retoques, que son 40 horas. Por lo que finalmente tenemos 536 + 10 + 40 que dan un total de 586 horas dentro de la empresa. A ésto le hemos de sumar mis horas dedicadas al TFG fuera de la empresa

donde se junta la memoria y la presentación, que aproximadamente serán de 40 horas, por lo que finalmente, en este TFG se usarán 626 horas.

Considero que la aplicación está acabada en cuanto a las funcionalidades. Sin embargo, me ha faltado tiempo para realizar los tests de rendimiento según comenté en los requisitos no funcionales de la memoria.

14.2. Proyecto de la empresa

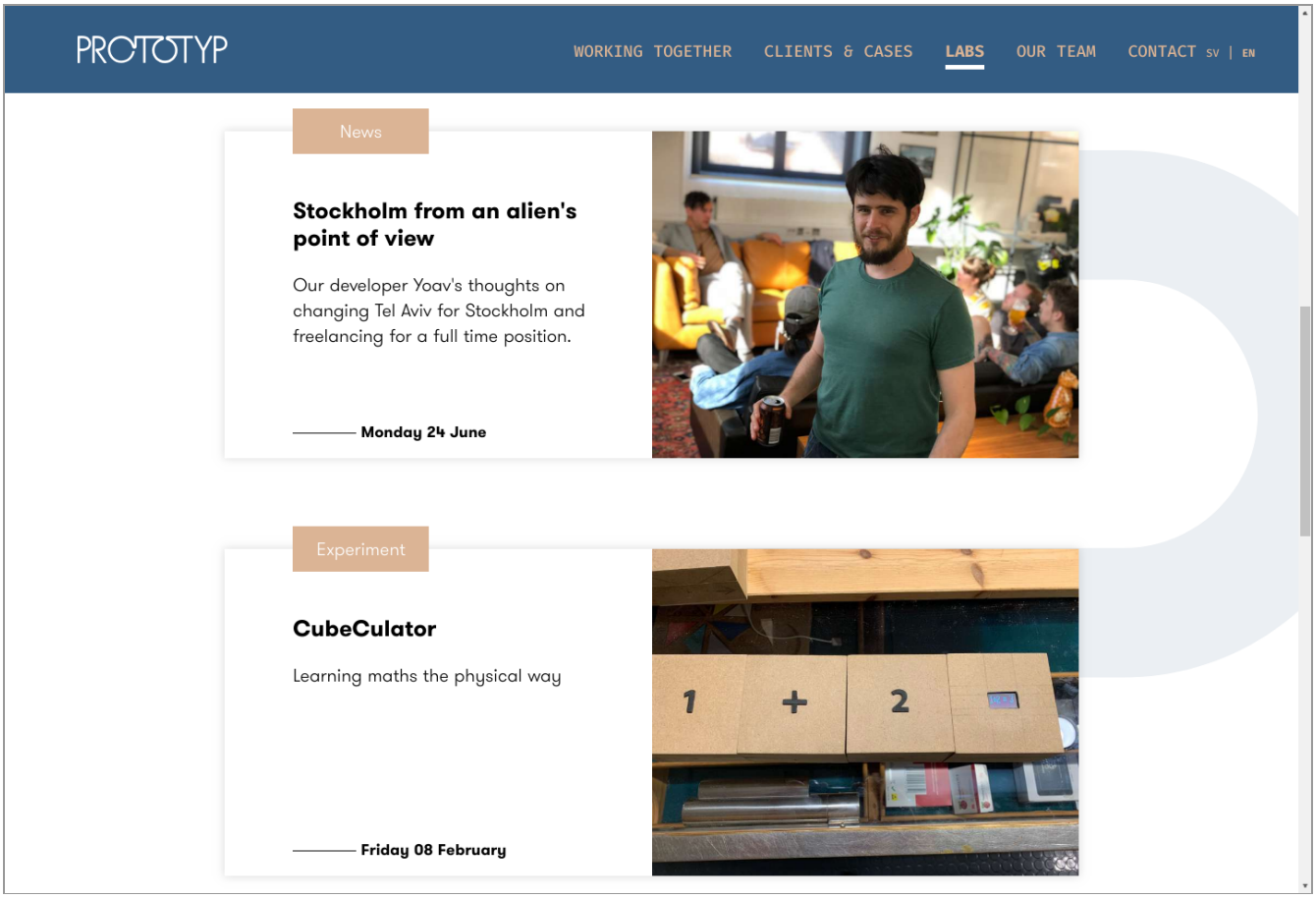


Imagen 90. Vista de lista de artículos del proyecto de la empresa.

Tal y como comento en el apartado anterior, he tenido que participar en un proyecto de la empresa en la que he realizado este TFG. El proyecto era desarrollar la aplicación web de la empresa, desde donde se pueden consultar trabajos hechos por la empresa, clientes, artículos tecnológicos, ver varias animaciones y demás.

Esta vez no he estado solo en el proyecto, sino que he tenido un compañero, de manera que aplicando una metodología como la explicada en el capítulo de aplicación de la metodología hemos conseguido acabar gran parte del proyecto en un mes. Además, el diseño front-end venía dado por un equipo de diseñadores, tanto la versión móvil como escritorio.

Este proyecto era más sencillo que el explicado en éste TFG. Mi compañero y yo escogimos la tecnología Vue para el front-end, que la comento en la sección de Frameworks valorados del capítulo de Implementación. Para el back-end, el director de la empresa nos hizo utilizar un 'headless CMS'[45], de manera que hemos construido una API que se consume por el cliente. La arquitectura es muy sencilla, no hay llamadas a servicios de terceros a excepción del headless CMS para obtener los datos.

Este pequeño proyecto ha sido más un trabajo de front-end, en el que he aprendido detalles que se han de tener en cuenta detalles a la hora de entregar un producto como un a aplicación web al cliente y que quizás no he profundizado tanto en éste TFG.

Cosas aprendidas:

- Server side rendering: En nuestro TFG, al ser una aplicación web, el contenido se renderiza en el mismo cliente. Mientras que con server side rendering, la web se renderiza en el servidor que almacena el proyecto front-end, de manera que es indexado de forma más rápida y eficiente por motores de búsqueda como Google. A largo plazo, la idea es que mi proyecto de TFG se expanda para que haya más clientes, por lo que hubiese sido buena idea tener ésto en mente.

- Metadatos: Son datos que se usan para indicar información a los motores de búsqueda. En el caso del proyecto de la empresa, hay diversos artículos en los que se han generado metadatos para que puedan ser mejor indexados en los buscadores. De ésta manera mejoramos los textos que salen en los motores de búsqueda para hacerlos más intuitivos de cara al cliente.

- Comprobar diferentes versiones de navegadores: Mi TFG sólo ha sido probado con el navegador Chrome, sin embargo, una aplicación web puede verse diferente según qué navegador web se utilice, ya que no todos los navegadores interpretan todas propiedades de los estilos CSS. Actualmente cada vez hay más compatibilidad entre éstos, pero por ejemplo, en el proyecto de la empresa utilicé un servicio llamado BrowserStack[46] que permite visualizar una web a través de diversos dispositivos y navegadores, se puede ver errores como la no aplicación de fuentes correctas en dispositivos Apple con el navegador Safari, diseño roto con Internet Explorer, etc.

14.3. Competencias técnicas

En este TFG se han marcado y conseguido las siguientes competencias técnicas:

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos.

[45] Un CMS es un gestor de contenido en el que viene incluido la capa de presentación, mientras que en un headless CMS, está hecho para crear una API que sea consumida por un cliente. [https://en.wikipedia.org/wiki/Headless_content_management_system]
[46] https://www.browserstack.com/

En este TFG se ha desarrollado y evaluado un servicio de software complejo, ya que está compuesto por diversos servicios de terceros donde uno de ellos trata con un sistema de pagos en línea.

CES1.2: Dar soluciones a problemas de integración en función de les estratégias, de los estándares y de las tecnologías disponibles.

Se han utilizado estrategias aprendidas a lo largo de la carrera, un ejemplo sería la arquitectura en tres capas utilizada en el proyecto. Gran parte del contenido de esta memoria está basado en mis conocimientos adquiridos a lo largo de la carrera, las decisiones escogidas para realizar éste proyecto no han sido arbitrarias, si no que hay una base sólida detrás.

CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que se puedan presentar.

Esta competencia técnica ha estado presente de forma implícita durante el desarrollo del proyecto. En el proyecto del servidor front-end se hacen validaciones de los datos introducidos de manera que el servidor no reciba datos erróneos. Pero aún así también se ha puesto una capa de validación en el proyecto del servidor back-end, de manera que se tengan controlados los posibles errores que surjan en la aplicación. También se toman precauciones con el sistema de pagos en línea usado.

CES1.5: Especificar, diseñar, implementar y evaluar bases de dades.

Para almacenar los datos de la aplicación se ha especificado y diseñado un modelo de datos para posteriormente implementarlo en forma de base de datos.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software.

Al principio del proyecto se especificó un requisito no funcional de mantenimiento que ha sido respetado. Se han incluido diversos tipos de pruebas para controlar el software construido.

CES2.1: Definir y gestionar los requisitos de un sistema software.

En las primeras semanas se han establecidos requisitos funcionales en forma de historias de usuario y requisitos no funcionales. Se han priorizado los requisitos funcionales que más peso dan a la aplicación durante el desarrollo para tener el núcleo del proyecto lo antes posible.

CES2.2: Diseñar soluciones apropiadas en uno o más dominios de la aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos.

En este proyecto se han tenido en cuenta aspectos legales en su capítulo correspondiente de la memoria. Además, el proyecto utiliza un sistema de pagos en línea integrado en el proyecto por lo que también tocamos aspectos económicos.

14.4. Integración de conocimientos

Para realizar este proyecto se ha hecho uso de los conocimientos adquiridos a través de las diversas asignaturas que he tenido en la FIB. A continuación enseño una lista de las principales y su cabida en el proyecto:

- **BASES DE DADES [BD]:** Me ha servido para el diseño de la base de datos del proyecto. He aprovechado los conceptos generales y utilizado herramientas similares a las que aprendí en la asignatura.

- **EMPRESA I ENTORN ECONÒMIC [EEE]:** He utilizado conceptos generales para realizar la parte económica de la entrega GEP (que también está incluida en la memoria).

- **INTRODUCCIÓ A L'ENGINYERIA DEL SOFTWARE [IES]:** Me ha servido para especificar y diseñar el sistema.

- **PROJECTES DE PROGRAMACIÓ [PROP]:** He utilizado las prácticas que utilicé en esta asignatura como por ejemplo el control de código con Git/Github.

- **INTERACCIÓ I DISSENY D'INTERFÍCIES [IDI]:** Me ha servido para tener en cuenta la interfaz gráfica de la aplicación construida. También para poder especificar los requisitos no funcionales de apariencia.

- **ARQUITECTURA DEL SOFTWARE [AS]:** He hecho uso de la documentación de esta asignatura para escribir la parte del diseño del sistema de la memoria.

- **ENGINYERIA DE REQUISITS [ER]:** He utilizado documentación y proyectos realizados en el transcurso de esta asignatura para definir los requisitos del sistema.

- **GESTIÓ DE PROJECTES DE SOFTWARE [GPS]:** Me ha servido para construir una metodología de trabajo. Además de conocer diversas herramientas útiles como Jira o Heroku.

- **APLICACIONS I SERVEIS WEB [ASW]:** De esta asignatura he extraído los conocimientos sobre la comunicación cliente - servidor de las aplicaciones web similares a la mía. De hecho, aquí aprendí las bases del framework MVC Ruby on Rails, ya que lo usé en el proyecto final de la asignatura.

- **PROJECTE D'ENGINYERIA DEL SOFTWARE [PES]:** El proyecto realizado en esta asignatura ha sido como una guía para el desarrollo de la aplicación de mi TFG, ya que también se ha desarrollado con la metodología ágil y se ha construido una aplicación (aunque móvil) que se comunica con un servidor.

14.5. Reflexión personal

Realizar este TFG ha sido una gran fuente de conocimiento. He podido aplicar muchos conceptos aprendidos durante la carrera y he podido adaptarme a las tecnologías actuales.

Además, el hecho de haberlo realizado en una empresa ha hecho que pueda aprender muchas cosas del mucho del desarrollo web, ya que no sólo han sido las reuniones que he tenido, sino el hecho de oír cosas interesantes día tras día en la oficina.

Respecto a los objetivos iniciales del trabajo. Se ha conseguido una aplicación web visualmente atractiva, funcional y que resuelve la formulación del problema inicial.

Finalmente, decir que lo que más me ha gustado del proyecto ha sido utilizar una metodología ágil flexible, ya que ha hecho que el proyecto no haya estado limitado al principio, sino que haya ido evolucionando en el tiempo, centrándome en el núcleo del producto.

15. Referencias

[1] - “La Sociedad De Consumo y Su Impacto Ambiental En El Planeta.” Sostenibilidad Para Todos, 21 Marzo 2018, www.sostenibilidad.com/desarrollo-sostenible/sociedad-consumo-impacto-ambiental-planeta/

[2] - Tena, María. “Metodología 'Agile'. La Revolución De Las Formas De Trabajo.” BBVA NOTICIAS, BBVA, 28 Nov. 2018, www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trabajo/

[3] - “What Is Agile Methodology?” Agile Methodologies for Software Development, 15 Jan. 2019, https://resources.collab.net/agile-101/agile-methodologies/

[4] - Verheyen, Gunther. Scrum - a Pocket Guide: a Smart Travel Companion. Van Haren Publ., 2013.

[5] - “Agile Project Management.” Agile Project Management, www.pivotaltracker.com/, 5 Feb. 2019,

[6] - “I Don't Speak Your Language: Frontend vs. Backend.” Treehouse Blog, 19 Dec. 2016, https://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend/

[7] - “Diagrama De Gantt.” Wikipedia, Wikimedia Foundation, 5 Feb. 2019, https://es.wikipedia.org/wiki/Diagrama_de_Gantt/

[8] - “Estudios De Remuneración.” Michael Page, 10 Feb. 2019, www.michaelpage.es/prensa-estudios/estudios/estudios-de-remuneracion/

[9] - “Billetes Sencillos y Abonos Integrados.” Hola Barcelona Travel Card | Transports Metropolitans De Barcelona, 11 Marzo 2019, www.tmb.cat/es/tarifas-metro-bus-barcelona/sencillos-e-integrados/

[10] - “Requisito (Sistemas).” Wikipedia, Wikimedia Foundation, 14 Feb. 2019, https://es.wikipedia.org/wiki/Requisito_(sistemas).

[11] - Robertson, James, and Suzanne Robertson. *Volere Requirements Specification Template*. 18th ed., The Atlantic Systems Guild Limited, 2016, Accessed 3 Apr 2019.

[12] - “Reglamento General De Protección De Datos.” Wikipedia, Wikimedia Foundation, 28 Mar. 2019, https://es.wikipedia.org/wiki/Reglamento_General_de_Protección_de_Datos.

[13] - Bahit, Eugenia. *Scrum y eXtreme Programming para Programadores*, 2012. Universidad Miguel Hernández. Web. 2 April 2019.

[14] - “Lenguaje Unificado De Modelado.” Wikipedia, Wikimedia Foundation, 25 Jan. 2019, https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado.

[15] - *Especificació en UML: Esquema conceptual de les dades*. Departament D'enginyeria De Serveis I Sistemes D'informació, Accessed 5 Apr 2019.

[16] - “Programación Por Capas.” Wikipedia, Wikimedia Foundation, 3 Dec. 2018, https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas.

[17] - “Concepto De HTTP.” Concepto, https://concepto.de/http.

[18] - Merino, Marcos. “¿Qué Es Una API y Para Qué Sirve?” TICbeat, 11 July 2014, https://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve.

[19] - “CRUD.” Wikipedia, Wikimedia Foundation, 23 Jan. 2019, https://es.wikipedia.org/wiki/CRUD.

[20] - “Elements Marketplace: Heroku Postgres.” Heroku Elements, https://elements.heroku.com/addons/heroku-postgresql.

[21] - *Patró Domain Model: de l’esquema d’especificació al de disseny.* Departament D'enginyeria De Serveis I Sistemes D'informació, Accessed 10 Apr 2019.

[22] - Ordóñez, Jonathan. “¿Qué Es Una API REST?” Idento, 18 Sept. 2018, https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest.

[23] - Ribas, Ester. “Qué es Api Rest y por qué debes de integrarla en tu negocio” Blog De IEBSchool, 29 May 2018, https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech.

[24] - “Guides.rubyonrails.org.” Action Controller Overview - Ruby on Rails Guides, https://guides.rubyonrails.org/action_controller_overview.html.

[25] - co2ok. “CO2ok For Woocommerce.” WordPress.org, https://wordpress.org/plugins/co2ok-for-woocommerce.

[26] - “AWS | Almacenamiento De Datos Seguro En La Nube (S3).” Amazon, Amazon, https://aws.amazon.com/es/s3.

[27] - UrbanInstitute. “UrbanInstitute/Quick-Quiz.” GitHub, 12 Mar. 2019, https://github.com/UrbanInstitute/quick-quiz.

[28] - “Active Admin.” Active Admin | The Administration Framework for Ruby on Rails, https://activeadmin.info/index.html.

[29] - Plataformatec. “Plataformatec/Devise.” GitHub, 5 Apr. 2019, https://github.com/plataformatec/devise.

[30] - Activeadmin-Plugins. “Activeadmin-Plugins/active_admin_theme.” GitHub, 12 Jan. 2019, https://github.com/activeadmin-plugins/active_admin_theme.

[31] - Pdfkit. “Pdfkit/Pdfkit.” GitHub, 25 Feb. 2019, https://github.com/pdfkit/pdfkit.

[32] - JMAlarcon. “¿Qué Es Un ORM?” CampusMVP.es, https://www.campusmvp.es/recursos/post/que-es-un-orm.aspx.

[33] - “Guides.rubyonrails.org.” Active Record Basics - Ruby on Rails Guides, https://guides.rubyonrails.org/active_record_basics.html.

[34] - “Guides.rubyonrails.org.” Active Record Migrations - Ruby on Rails Guides, https://guides.rubyonrails.org/active_record_migrations.html.

[35] - “AngularJS.” Wikipedia, Wikimedia Foundation, 15 Apr. 2019, https://es.wikipedia.org/wiki/AngularJS.

[36] - “Pros and Cons of Vue.js Framework.” Agriya Blog, 15 May 2017, https://www.agriya.com/blog/pros-and-cons-of-vue-js-framework.

[37] - Barros, Afonso. “Angular / React / Vue Pros and Cons.” Medium, Medium, 11 Apr. 2018, https://medium.com/@afonsobarros/angular-react-vue-pros-and-cons-75e161311e86.

[38] - Tutorialspoint.com. “CSS3 Responsive.” Www.tutorialspoint.com, https://www.tutorialspoint.com/css/css_responsive.htm.

[39] - Pallerols, Xavier Martí. “Qué es el responsive design y por qué tu web debería serlo.” Blog De IEBSchool, 23 Nov. 2017, https://www.iebschool.com/blog/que-es-responsive-web-design-analitica-usabilidad.

[40] - “Breakpoints - Material-UI.” Material, https://material-ui.com/layout/breakpoints.

[41] - “Customize Checkout Button.” Customize Checkout Button - PayPal Developer, https://developer.paypal.com/docs/partners/checkout/customize-checkout-button/?mark=styl.

[42] - DjangoGirls. “¿Qué Es Django?” Django Girls Tutorial, https://tutorial.djangogirls.org/es/django/.

[43] - “WebStorm Reviews: Overview, Pricing and Features.” https://reviews.financesonline.com/p/webstorm/.

[44] - “Integración Continua.” Wikipedia, Wikimedia Foundation, 23 Apr. 2019, https://es.wikipedia.org/wiki/Integraci%C3%B3n_continua/.

[45] - Atlassian. “Git Workflow | Atlassian Git Tutorial.” Atlassian, 24 May, 2019, http://es.atlassian.com/git/tutorials/comparing-workflows/

[46] - “GitHub.” Wikipedia, Wikimedia Foundation, 7 June 2019, https://es.wikipedia.org/wiki/GitHub/

[47] - Ricardocelis. “¿Qué Es Heroku y Para Qué Me Sirve?” Platzi.com, Platzi, 3 Nov. 2017, https://platzi.com/blog/que-es-heroku-y-para-que-me-sirve/

[48] - Herranz, Jose Ignacio. “TDD Como Metodología De Diseño De Software.” Paradigma, 8 Aug. 2017, https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/.

[49] - Thoughtbot. “Thoughtbot/Shoulda-Matchers.” GitHub, 13 June 2019, https://github.com/thoughtbot/shoulda-matchers/.

[50] - “Ley Orgánica De Protección De Datos De Carácter Personal (España).” Wikipedia, Wikimedia Foundation, 28 May 2019, https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_de_Car%C3%A1cter_Personal_(Espa%C3%B1a)/

[51] - Andrearrs. “Cómo Elegir Licencias Open Source Para Tu Proyecto.” Hipertextual, 22 May 2014, https://hipertextual.com/archivo/2014/05/como-elegir-licencias-open-source/

[52] - “Cookie (Informática).” Wikipedia, Wikimedia Foundation, 24 May 2019, https://es.wikipedia.org/wiki/Cookie_(inform%C3%A1tica)/